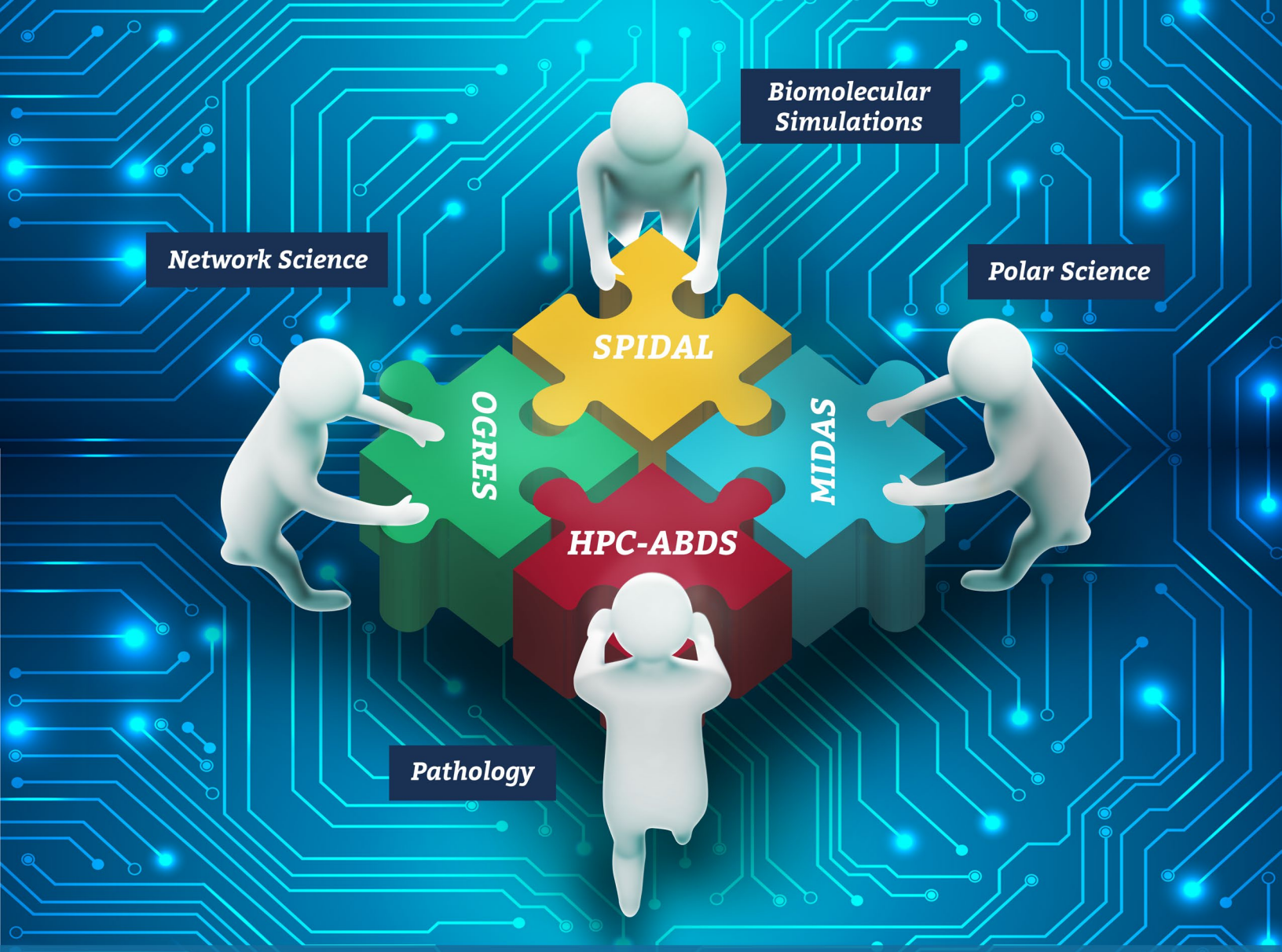


DATANET: CIF21 DIBBS: Middleware and High Performance Analytics Libraries for Scalable Data Science

Progress Report August 2016

Indiana University (Fox, Qiu, Crandall, von Laszewski), Rutgers (Jha), Virginia Tech (Marathe), Kansas (Paden), Stony Brook (Wang), Arizona State (Beckstein), Utah (Cheatham)

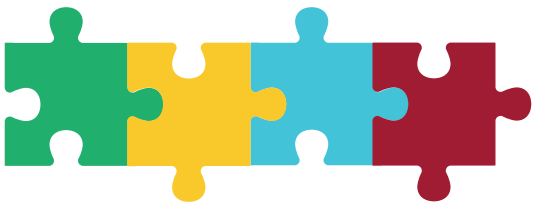
Datanet: CIF21 DIBBs: Middleware and High Performance Analytics Libraries for Scalable Data Science
NSF14-43054 Progress Report August 2016

Indiana University (Fox, Qiu, Crandall, von Laszewski)
Rutgers (Jha)
Virginia Tech (Marathe)
Kansas (Paden)
Stony Brook (Wang)
Arizona State (Beckstein)
Utah (Cheatham)

Table of Contents

1.	**Introduction**	**4**
1.1.	Preamble	4
1.2.	Motivation	5
1.3.	Overview of Components of SPIDAL Dibbs	6
2.	**Overall Architecture**	**8**
2.1.	NIST Big Data Application Analysis	8
2.2.	HPC-ABDS High Performance Computing and Apache Big Data Stack	11
2.3.	Big Data - Big Simulation (Exascale) Convergence	12
2.4.	HPC-ABDS Mapping of Project Activities	13
3.	**MIDAS: Software Activities in DIBBS**	**14**
3.1.	Introduction	14
3.2.	SPIDAL Language	15
3.3.	Cloudmesh Interoperability IaaS and Paas Tool leveraging DevOps	20
3.4.	Pilot Jobs and Pilot Data Memory	23
3.5.	Architecture of Scalable Big Data Machine Learning Library	26
3.6.	Harp Programming Paradigm	29
3.7.	Integration of Harp and Intel DAAL Library	32
4.	**SPIDAL Algorithms**	**34**
4.1.	Introduction	35
4.2.	SPIDAL Algorithms – Harp Latent Dirichlet Allocation	37
4.3.	SPIDAL Algorithms – Subgraph mining	38
4.4.	SPIDAL Algorithms – Random Graph Generation	41
4.5.	SPIDAL Algorithms – Triangle Counting	43
4.6.	SPIDAL Algorithms – Community Detection	44
4.7.	SPIDAL Algorithms – Core	45
4.8.	SPIDAL Algorithms – Optimization	48
4.9.	SPIDAL Algorithms – Polar Remote Sensing Algorithms	50
4.10.	SPIDAL Algorithms – Nuclei Segmentation for Pathology Images	53
4.11.	SPIDAL Algorithms – Spatial Querying Methods	56

5.	**Applications**	**58**
5.1.	Summary	58
5.2.	Overview of Imaging Applications	59
5.3.	Enabled Applications – Digital Pathology	60
5.4.	Enabled Applications – Public Health	61
5.5.	Enabled Applications - Biomolecular Simulation Data Analysis	62
6.	**Community Engagement**	**68**
6.1.	REU Programs	68
6.2.	Making MIDAS and SPIDAL Available to community	68
6.3.	Working with Apache: Harp and Heron	69
7.	**Futures**	**71**
7.1.	Integrating SPIDAL and MIDAS as Coherent Building Blocks	71
7.2.	Orchestration and Workflow	72
7.3.	Streaming	72
8.	**Team & Publications**	**74**
9.	**References**	**77**



1 Introduction

1.1. Preamble

1.2 Motivation

1.3 Overview of Components of SPIDAL Dibbs

1.1. Preamble

This is a 21-month progress report on an NSF-funded project NSF14-43054 started October 1, 2014 and involving a collaboration between university teams at Arizona, Emory, Indiana (lead), Kansas, Rutgers, Virginia Tech, and Utah. The project is constructing data building blocks to address major cyberinfrastructure challenges in seven different communities: Biomolecular Simulations, Network and Computational Social Science, Epidemiology, Computer Vision, Spatial Geographical Information Systems, Remote Sensing for Polar Science, and Pathology Informatics.

The project has an overall architecture [5] built around the twin concepts of HPC-ABDS (High Performance Computing enhanced Apache Big Data Stack) software [6-8] and a classification of Big data applications – the Ogres [9-11] – that defined the key qualities exhibited by applications and required to be supported in software. These underpinning ideas are described in section 2 together with recent extensions including a discussion of Big Data – Big Simulation and HPC convergence [12, 13].

Our architecture for data intensive applications relies on Apache Big Data stack ABDS for the core software building blocks where we add an interface layer MIDAS – the Middleware for Data-Intensive Analytics and Science – described in section 3, that will enable scalable applications with the performance of HPC (High Performance Computing) and the rich functionality of the commodity ABDS (Apache Big Data Stack). Then the next set of building blocks described in Section 4 are members of a cross-cutting high-performance data-analysis library -- SPIDAL (Scalable Parallel Interoperable Data Analytics Library). SPIDAL consists of a set of core routines covering well established functionality (such as optimization and clustering) together with targeted community specific capabilities motivated by applications described in section 5. Section 6 describes community engagement and section 7 has some important lessons learnt and existing and future spin-off activities.

The project has a web page [14], an early Indiana University press release [15] and the public NSF award announcement [16].

1.2. Motivation

Many scientific problems depend on the ability to analyze and compute on large amounts of data. This analysis often does not scale well; its effectiveness is hampered by the increasing volume, variety and rate of change (velocity) of big data. This project is designing, developing and implementing building blocks that will enable a fundamental improvement in the ability to support data intensive analysis on a broad range of cyberinfrastructure, including that supported by NSF for the scientific community.

The project will integrate features of traditional high-performance computing, such as scientific libraries, communication and resource management middleware, with the rich set of capabilities found in the commercial Big Data ecosystem. The latter includes many important software systems such as Hadoop, Spark, Storm and Mesos, available from the Apache open source community. We note that there are over 350 separate software modules in HPC-ABDS [7] and it is certainly not realistic to study use, and/or support this number in, for example, the national NSF cyberinfrastructure. This project divides this software into broad categories and identifies a few key or representative members whose performance is examined and enhanced by HPC.

We are inspired by the beneficial impact that scientific libraries such as PETSc, MPI and ScaLAPACK have had for supercomputer simulations and hope that our building blocks MIDAS and SPIDAL will have the same impact on data analytics. MIDAS will allow our libraries to be scalable

and interoperable across a range of computing systems including clouds, clusters and supercomputers.

1.3. Overview of Components of SPIDAL DIBBs

The implementation of this project requires significant coordinated activity in several areas that are spelled out here and described in more detail in the later sections

- **NIST Big Data Application Analysis [10, 11, 17, 18]**– This identifies features of data intensive applications that need to be supported in software and represented in benchmarks. This analysis comes from the project (initiated as part of proposal planning) and was recently extended to separately look at model and data components [13, 19]. (Section 2.1)
- **HPC-ABDS:** Cloud-HPC interoperable software **performance of HPC** (High Performance Computing) and the rich **functionality** of the commodity **Apache Big Data Stack**. [6, 7] It is described in Section 2.2.
    - This is a reservoir of software subsystems – nearly all from outside the project and coming from a mix of HPC and Big Data communities.
    - We added a categorization and an HPC enhancement approach
    - HPC-ABDS combined with the NIST Big Data Application Analysis leads to Big Data – Big Simulation – HPC Convergence [12, 13], described in Section 2.3
- **MIDAS:** This is the integrating Middleware that links HPC and ABDS: its different components are described in Section 3. It includes an architecture for Big data analytics, a cloud-HPC interoperable deployment tool, and other features,
- **SPIDAL Java:** Our goals imply a substantial emphasis on performance of MIDAS Inter- and Intra-node. This is extended to include the techniques described in Section 3.2, to achieve high performance when coding in the popular data language Java.
- **SPIDAL (Scalable Parallel Interoperable Data Analytics Library):** This is described in Section 4 and provides scalable data analytics for:
    - Domain specific data analytics libraries – mainly from project.
    - Add Core Machine learning libraries – mainly from community.
- **Benchmarks** – This project adds to a community with Ogre characteristics and high performance core kernels as discussed at WBDB2015 Benchmarking Workshop.

- **Implementations**: We have a particular focus on NSF infrastructure including XSEDE and Blue Waters as well as clouds with OpenStack and Docker using Cloudmesh (Section 3.3) for interoperability.
- **Applications:** Biomolecular Simulations, Network and Computational Social Science, Epidemiology, Computer Vision, Spatial Geographical Information Systems, Remote Sensing for Polar Science and Pathology Informatics. These are described in Section 5 and provide requirements and test grounds for our work. Separately funded work in bioinformatics and computer vision also help in these regards.

2 Overall Architecture

2.1. NIST Big Data Application Analysis

2.2. HPC-ABDS High Performance Computing and Apache Big Data Stack

2.3. Big Data - Big Simulation (Exascale) Convergence

2.4. HPC-ABDS Mapping of Project Activitie

2.1. NIST Big Data Application Analysis

The Big Data Ogres build on a collection of 51 big data uses gathered by the NIST Public Working Group where 26 properties were gathered for each application [20]. This information was combined with other studies including the Berkeley dwarfs [21], the NAS parallel benchmarks [22, 23] and the Computational Giants of the NRC Massive Data Analysis Report [24]. The Ogre analysis led to a set of 50 features divided into four views that could be used to categorize and distinguish between applications. The four views are Problem Architecture (Macro pattern); Execution Features (Micro patterns); Data Source and Style; and finally the Processing View or runtime features. We generalized [3] this approach to integrate Big Data and Simulation applications into a single classification that we called convergence diamonds with the total facets growing to 64 in number and split between the same 4 views as shown in figure 2-1. These are used in Section 2.3 and a mapping of facets into the work of this project as given earlier[11].

We can illustrate these facets by considering a few of special applicability to our project. The facets in the Problem Architecture view include 5 very common ones describing synchronization structure of a parallel job: Pleasingly Parallel (PA1), MapReduce (PA2), MapCollective (PA3) and Map Point-to-Point (PA4) describe respectively the processing of a collection of independent events; independent calculations (maps) followed by a final consolidation via MapReduce; parallel machine learning dominated by scatter, gather, reduce and broadcast; simulations or graph processing with many local linkages in points of studied system. The fifth important Problem Architecture is Map Streaming (PA5) seen in recent approaches to processing real-time data [25]. We do not focus on pure shared memory architectures PA-6 although we do look carefully at hybrid architectures with clusters of multicore nodes and find important performances differences dependent on the node programming model (section 3.2). Most of our codes are SPMD (PA-7) and BSP (PA-8).

Looking at the Execution View we see in EV-M14 complexity of model (O(N2) for N points) seen in the non-metric space models EV-M13 such as one gets with DNA sequences. EV-M11 describes iterative structure distinguishing Spark, Flink, and Harp from the original Hadoop. The facet EV-M8 describes the communication structure which is a focus of our research as much data analytics relies on collective communication which is in principle understood but we find that significant new work is needed compared to basic MPI releases. The model size EV-M4 and data volume EV-D4 are important in describing the algorithm performance as just like in simulation problems, the grain size (the number of model parameters held in the unit – thread or process – of parallel computing) is a critical measure of performance.

In the Data view, we can highlight D-5 streaming covered in section 7.1 where there is a lot of recent progress; D-9 categorizes our Biomolecular simulation application with data produced by an HPC simulation; and D-10 Geospatial Information Systems covered by spatial algorithms in Section 4.11. D-7 (provenance) is an example of an important feature that we are not covering. The data storage and access D-3 and D-4 is covered in section 3-4. The Internet of Things D-8 is not a focus of our project although our recent streaming work (section 7.1) relates to this and our addition of HPC to Apache Heron and Storm is an example of the value of HPC-ABDS to IoT.

The Processing View characterizes algorithms with for example, Graph P-M13 (sections 4.2, 3.7, 4.4) and visualization P-M14 covered in Section 4.5. P-M15 directly describes SPIDAL which is a library of core and other analytics. This project covers many aspects of P-M4 to P-M11 as these characterize the SPIDAL algorithms. We are of course NOT addressing P-M16 to P-M22 which are simulation algorithms and not applicable to data analytics. Our work largely addresses Global Machine Learning P-M3 although some of the simple image analytics are local machine learning P-M2 with parallelism over images and not over the analytics. Many of our SPIDAL algorithms have

linear algebra at their core; one nice example is multi-dimensional scaling in section 4.8(2) which is based on matrix-matrix multiplication.

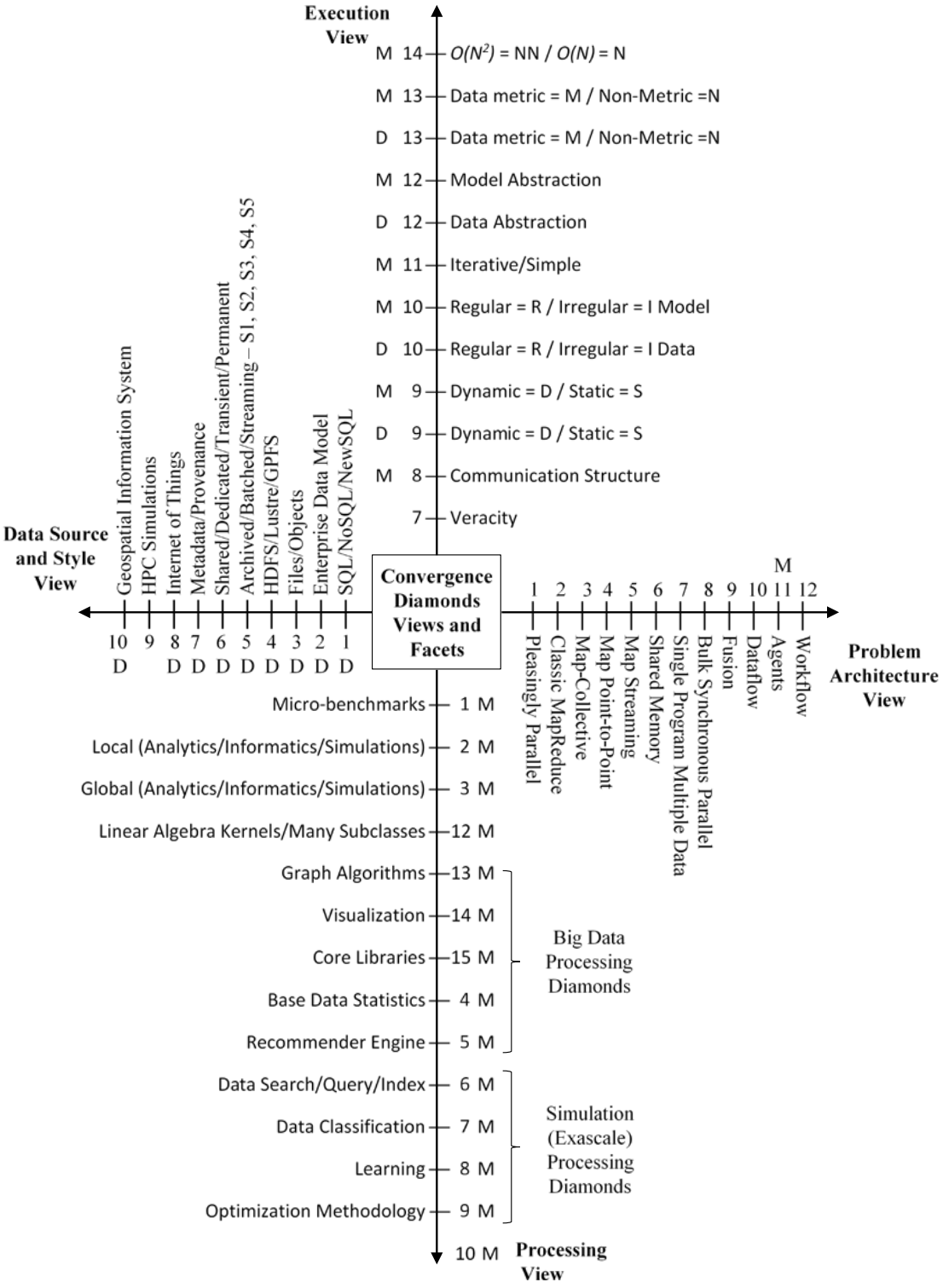


Figure 2-1: 64 Convergence Diamonds [12]in 4 views generalizing Ogres.

These convergence diamonds are particularly useful in classifying benchmarks and we are pursuing this to design a "complete" (over the 64 facets) set of benchmarks which potentially will link simulation and big data benchmark collections.

2.2. HPC-ABDS High Performance Computing and Apache Big Data Stack

Kaleidoscope of (Apache) Big Data Stack (ABDS) and HPC Technologies	
Cross-Cutting Functions	**17) Workflow-Orchestration:** ODE, ActiveBPEL, Airavata, Pegasus, Kepler, Swift, Taverna, Triana, Trident, BioKepler, Galaxy, IPython, Dryad, Naiad, Oozie, Tez, Google FlumeJava, Crunch, Cascading, Scalding, e-Science Central, Azure Data Factory, Google Cloud Dataflow, NiFi (NSA), Jitterbit, Talend, Pentaho, Apatar, Docker Compose, KeystoneML
1) **Message and Data Protocols:** Avro, Thrift, Protobuf	**16) Application and Analytics:** Mahout , MLlib , MLbase, DataFu, R, pbdR, Bioconductor, ImageJ, OpenCV, Scalapack, PetSc, PLASMA MAGMA, Azure Machine Learning, Google Prediction API & Translation API, mlpy, scikit-learn, PyBrain, CompLearn, DAAL(Intel), Caffe, Torch, Theano, DL4j, H2O, IBM Watson, Oracle PGX, GraphLab, GraphX, IBM System G, GraphBuilder(Intel), TinkerPop, Parasol, Dream:Lab, Google Fusion Tables, CINET, NWB, Elasticsearch, Kibana, Logstash, Graylog, Splunk, Tableau, D3.js, three.js, Potree, DC.js, TensorFlow, CNTK
2) **Distributed Coordination:** Google Chubby, Zookeeper, Giraffe, JGroups	**15B) Application Hosting Frameworks:** Google App Engine, AppScale, Red Hat OpenShift, Heroku, Aerobatic, AWS Elastic Beanstalk, Azure, Cloud Foundry, Pivotal, IBM BlueMix, Ninefold, Jelastic, Stackato, appfog, CloudBees, Engine Yard, CloudControl, dotCloud, Dokku, OSGi, HUBzero, OODT, Agave, Atmosphere **15A) High level Programming:** Kite, Hive, HCatalog, Tajo, Shark, Phoenix, Impala, MRQL, SAP HANA, HadoopDB, PolyBase, Pivotal HD/Hawq, Presto, Google Dremel, Google BigQuery, Amazon Redshift, Drill, Kyoto Cabinet, Pig, Sawzall, Google Cloud DataFlow, Summingbird, Lumberyard
3) **Security & Privacy:** InCommon, Eduroam OpenStack Keystone, LDAP, Sentry, Sqrrl, OpenID, SAML OAuth	**14B) Streams:** Storm, S4, Samza, Granules, Neptune, Google MillWheel, Amazon Kinesis, LinkedIn, Twitter Heron, Databus, Facebook Puma/Ptail/Scribe/ODS, Azure Stream Analytics, Floe, Spark Streaming, Flink Streaming, DataTurbine **14A) Basic Programming model and runtime**, **SPMD, MapReduce:** Hadoop, Spark, Twister, MR-MPI, Stratosphere (Apache Flink), Reef, Disco, Hama, Giraph, Pregel, Pegasus, Ligra, GraphChi, Galois, Medusa-GPU, MapGraph, Totem
4) **Monitoring:** Ambari, Ganglia, Nagios, Inca	**13) Inter process communication Collectives, point-to-point, publish-subscribe:** MPI, HPX-5, Argo BEAST HPX-5 BEAST PULSAR, Harp, Netty, ZeroMQ, ActiveMQ, RabbitMQ, NaradaBrokering, QPid, Kafka, Kestrel, JMS, AMQP, Stomp, MQTT, Marionette Collective, **Public Cloud:** Amazon SNS, Lambda, Google Pub Sub, Azure Queues, Event Hubs
	12) In-memory databases/caches: Gora (general object from NoSQL), Memcached, Redis, LMDB (key value), Hazelcast, Ehcache, Infinispan, VoltDB, H-Store
	12) Object-relational mapping: Hibernate, OpenJPA, EclipseLink, DataNucleus, ODBC/JDBC
21 layers Over 350 Software Packages **January 29 2016** Green is work of NSF14-43054	**12) Extraction Tools:** UIMA, Tika
	11C) SQL(NewSQL): Oracle, DB2, SQL Server, SQLite, MySQL, PostgreSQL, CUBRID, Galera Cluster, SciDB, Rasdaman, Apache Derby, Pivotal Greenplum, Google Cloud SQL, Azure SQL, Amazon RDS, Google F1, IBM dashDB, N1QL, BlinkDB, Spark SQL
	11B) NoSQL: Lucene, Solr, Solandra, Voldemort, Riak, ZHT, Berkeley DB, Kyoto/Tokyo Cabinet, Tycoon, Tyrant, MongoDB, Espresso, CouchDB, Couchbase, IBM Cloudant, Pivotal Gemfire, HBase, Google Bigtable, LevelDB, Megastore and Spanner, Accumulo, Cassandra, RYA, Sqrrl, Neo4J, graphdb, Yarcdata, AllegroGraph, Blazegraph, Facebook Tao, Titan:db, Jena, Sesame **Public Cloud:** Azure Table, Amazon Dynamo, Google DataStore
	11A) File management: iRODS, NetCDF, CDF, HDF, OPeNDAP, FITS, RCFile, ORC, Parquet
	10) Data Transport: BitTorrent, HTTP, FTP, SSH, Globus Online (GridFTP), Flume, Sqoop, Pivotal GPLOAD/GPFDIST
	9) Cluster Resource Management: Mesos, Yarn, Helix, Llama, Google Omega, Facebook Corona, Celery, HTCondor, SGE, OpenPBS, Moab, Slurm, Torque, Globus Tools, Pilot Jobs
	8) File systems: HDFS, Swift, Haystack, f4, Cinder, Ceph, FUSE, Gluster, Lustre, GPFS, GFFS **Public Cloud:** Amazon S3, Azure Blob, Google Cloud Storage
	7) Interoperability: Libvirt, Libcloud, JClouds, TOSCA, OCCI, CDMI, Whirr, Saga, Genesis
	6) DevOps: Docker (Machine, Swarm), Puppet, Chef, Ansible, SaltStack, Boto, Cobbler, Xcat, Razor, CloudMesh, Juju, Foreman, OpenStack Heat, Sahara, Rocks, Cisco Intelligent Automation for Cloud, Ubuntu MaaS, Facebook Tupperware, AWS OpsWorks, OpenStack Ironic, Google Kubernetes, Buildstep, Gitreceive, OpenTOSCA, Winery, CloudML, Blueprints, Terraform, DevOpSlang, Any2Api
	5) IaaS Management from HPC to hypervisors: Xen, KVM, QEMU, Hyper-V, VirtualBox, OpenVZ, LXC, Linux-Vserver, OpenStack, OpenNebula, Eucalyptus, Nimbus, CloudStack, CoreOS, rkt, VMware ESXi, vSphere and vCloud, Amazon, Azure, Google and other public Clouds **Networking:** Google Cloud DNS, Amazon Route 53

Figure 2-2: HPC-ABDS as compiled January 29 2016 with layers given special consideration in this project shown in green.

Figure 2-2 collects together much existing relevant systems software coming from either HPC or commodity ABDS sources. The software is broken up into 21 layers so systems are grouped by functionality. The layers where there is especial attention paid in this project are colored green in Figure 2 and discussed in section 2.4. This software collection is termed HPC-ABDS (High Performance Computing enhanced Apache Big Data Stack) as many critical core components of the commodity stack (such as Spark and Hbase) come from open source projects while HPC is needed to bring performance and other parallel computing capabilities [6]. Note that Apache is the largest but not sole source of open source software; we believe that the Apache Foundation is a critical leader in the Big Data open source software movement and use it to designate the full big data software ecosystem. The figure also includes proprietary systems as they illustrate key capabilities and often motivate open source equivalents. We built this picture for big data problems but it also applies to big simulation with the caveat that we need to add more high level software at the library level and more high level tools like Global Arrays.

The essential idea of our Big Data HPC convergence for software is to make use of ABDS software where possible as it offers richness in functionality, a compelling open-source community sustainability model and typically attractive user interfaces. ABDS has a good reputation for scale but often does not give good performance. Our approach is to augment ABDS with HPC ideas which we illustrated with Hadoop [25, 26], Storm [27] and the basic Java environment [28]. As described in section 2.3, we suggest using the resultant HPC-ABDS for both big data and big simulation applications.

2.3. Big Data - Big Simulation (Exascale) Convergence

A key idea introduced in [12, 19] was to separate for any application, the data and model components which were merged together in the original Ogre analysis. In Big Data problems, naturally the data size is large and this normally is the focus of work in this area. However, a model is essential to interpret data and this is of course the concern of the rapid advances in machine learning and our SPIDAL library realizes the model algorithms. Note the size of a model can be much smaller than the data as in algorithms like clustering and dimension reduction. However, in applications like deep learning and topic modelling, the model can be huge. Parallelism has to be considered carefully both for data and model [19] and this leads to a new convergence programming paradigm. Turning to simulations where HPC has been most extensively explored, we again find that applications contain both data and model but typically it is now the model that is always large. For example, if one is solving particle dynamics or partial differential equations, then the model is the large numerical representation, while the data can be relatively small as in boundary conditions. On the other hand, there are examples like data assimilation for climate

forecasting and data visualizations produced by simulations, where the data can be quite large. The data is often static between iterations (unless streaming) while Model parameters vary between iterations.

We suggest that as long as one carefully compares apples with apples (e.g. Big Data model component with simulation model component), one can find many points of similarity between big data and simulations and see how to support both with a common architecture that is separate in the handling of the different data and model components but NOT separated by the application type: Big Data and Simulation.

2.4. HPC-ABDS Mapping of Project Activities

We can gain some insight into our project by mapping the work into the different levels of HPC-ABDS in Figure 2. The layers correspond to those colored green in figure 2.

- **Level 17: Orchestration**: Apache Beam (Google Cloud Dataflow) or Crunch integrated with Cloudmesh on HPC cluster
- **Level 16: Applications:** Datamining for molecular dynamics, Image processing for remote sensing and pathology, graphs, streaming, bioinformatics, social media, financial informatics, text mining
- **Level 16: Algorithms:** Generic and custom for applications **SPIDAL**
- **Level 14: Programming:** Storm, Heron (from Twitter replaces Storm), Hadoop, Spark, Flink. Improve Inter- and Intra-node performance
- **Level 13: Communication:** Enhanced Storm and Hadoop using HPC runtime technologies, Harp
- **Level 12: In-memory Database:** Redis and Spark used in Pilot-Data Memory
- **Level 11: Data management:** Hbase and MongoDB integrated via use of Beam and other Apache tools; enhance Hbase
- **Level 9: Cluster Management:** Integrate Pilot Jobs with Yarn, Mesos, Spark, Hadoop; integrate Storm and Heron with Slurm
- **Level 6: DevOps:** Python Cloudmesh virtual Cluster Interoperability

3 MIDAS: Software Activities in DIBBS

3.1. Introduction

3.2. SPIDAL Language

3.3. Cloudmesh Interoperability IaaS and Paas Tool leveraging DevOps

3.4. Pilot Jobs and Pilot Data Memory

3.5. Architecture of Scalable Big Data Machine Learning Library

3.6. Harp Programming Paradigm

3.7. Integration of Harp and Intel DAAL Library

3.1. Introduction

Recall from the introduction that MIDAS is the software shim that links HPC and ABDS. The initial algorithm work went ahead with traditional technologies and as MIDAS matures, it needs to be reworked with this infrastructure. The following subsections 3.2 to 3.6 expand section 2.4 and cover the different parts of MIDAS used in our research.

- **Section 3.2: SPIDAL Language** takes another look at Java Grande[29], 15 years after this was active, to examine how to make Java codes run as fast as possible with simple steps.

- **Section 3.3: The DevOps tool Cloudmesh** provides interoperability between HPC and Cloud (OpenStack, AWS, Docker) platforms based on virtual clusters with software defined systems using Ansible or Chef.

- **Section 3.4: Pilot Jobs** integrate Slurm with Yarn & Mesos (ABDS schedulers), and support ABDS programming frameworks (Hadoop MapReduce, Spark).

- **Section 3.4: Pilot Data Memory** integrates ABDS in-memory stores from Redis and Spark with HPC file systems

- **Section 3.5: Model-centric Data Analytics Architecture** provides a general approach to data analytics supporting different model synchronization approaches and with typically higher performance than parameter server approach

- **Section 3.6: Communication** and scientific **data abstractions**: Harp plug-in to Hadoop outperforms ABDS programming layers. Optimize collectives above that provided by MPI.

- **Data Management**: use Hbase, MongoDB with customization (pre-proposal work: no recent activity)

- **Workflow**: Use Apache Crunch and Beam (Google Cloud Data flow) as they link to other ABDS technologies. (just prototyping at present and no results reported)

We are starting to integrate MIDAS components and move into algorithms of SPIDAL Library.

3.2. SPIDAL Language

Motivation: SPIDAL Java was developed to support the high performance parallel multidimensional scaling and clustering algorithms listed in Section 4.5 and implemented in Java. These algorithms fall under the Map-Collective pattern, where independent computations are followed by global collective communications over many iterations. The goal of SPIDAL Java is to leverage HPC hardware in satisfying the demanding computation and communication needs of these algorithms. The work includes coding strategies and communication routines and allows Java to perform at a similar level to C++.

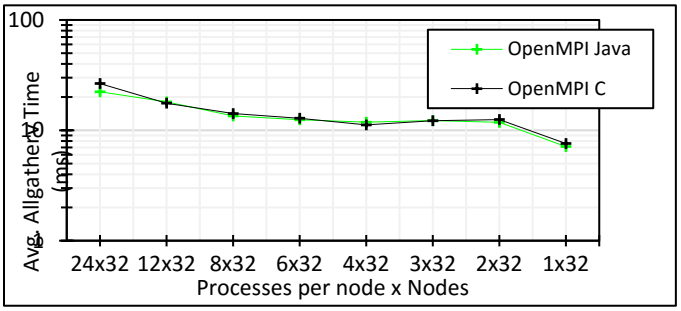


Figure 3-1 MPI allgatherv performance with varying processes per node

The nature of SPIDAL Java algorithms presents several challenges in coming up with high performance scalable and parallel implementations. Modern HPC clusters with multicore NUMA nodes provide a large number of computing cores. Intel Haswell, for example supports up to 24 and 36 core counts per node. The all MPI approach – 1 process per core on all nodes – is a straightforward match in

implementing Map-Collective applications on such clusters. The downside is the cost of collective communication, especially within a node. MPI+X model where X is a mechanism to do intra-node parallelism. Typically, scientific computations employ OpenMP as X.

While performance studies exist on the MPI+X model with regard to HPC applications, it is not clear what the performance characteristics would be for Java based Map-Collective applications. In SPIDAL Java, we identified that intra-node communication poses a significant cost, especially with collective communications. Figure 3 shows the effect of doing an *allgatherv* call with both Java and C versions of OpenMPI with varying number of processes per node. Note, the total number of bytes sent out from a node is constant across all the patterns in the x-axis.

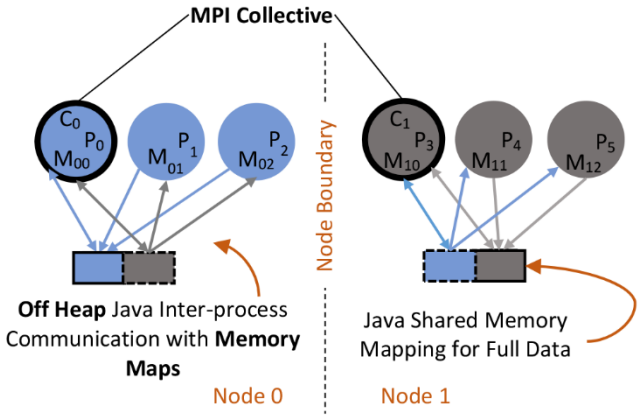


Figure 3-2 Intra-node message passing with Java shared memory maps

Figure 3-1 suggests keeping the number of communicating processes per node to a minimum gives the best performance. This implies intra-node parallelism should be done with threads. However, we find keeping the same process model, but doing intra-node communication through shared memory is better than threads (reasons explained below). In SPIDAL Java, a separate layer written in Java on top of MPI handles the intra-node communication through direct memory copies. The architecture is shown in Figure 3.

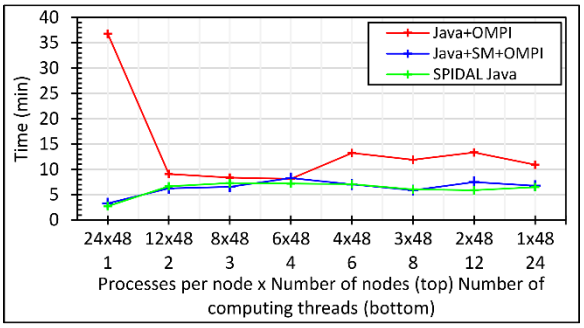


Figure 3-3 DA-MDS 100K performance with varying intra-node parallelism

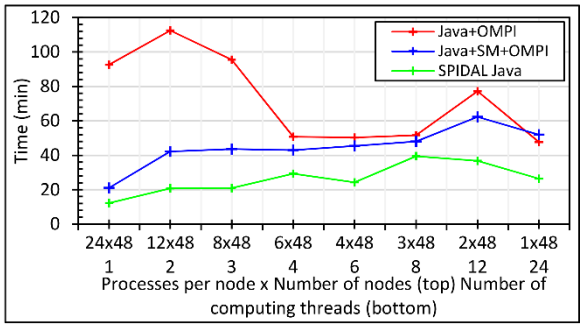


Figure 3-4 DA-MDS 200K performance with varying intra-node parallelism

Figures 3-3 and 3-4 show the results for 100K and 200K DA-MDS with (blue line) and without (red line) this optimization. The green line includes shared memory and several other optimizations available in SPIDAL Java.

The abscissa in Figure 3- and Figure 3- show different combinations of threads and processes within a node that utilizes the full parallelism of 24 cores per node. While typical MPI implementation (red line) favors threads within a node due to high communication cost, the removal of intra-node communication (blue line) shows all processes model (left most pattern) is better than other combinations that include threads. This is further exemplified in SPIDAL Java when other optimizations are applied on top of shared memory communication (see green lines).

Three reasons why threads don't perform as well as processes in these experiments are as follows.

1. NUMA Boundaries
   The experiments were run on a cluster where each node has 2 physical CPUs. When threads are scheduled across NUMA boundaries then accessing process local data could incur high overheads.

2. Scheduling Overhead
   Threads are used in a Fork-Join (FJ) pattern in these applications meaning that the worker threads sleep during serial paths of the code. We find through Intel Vtune profiling that the cost of scheduling these parallel regions over many iterations adds a significant overhead.

3. TLB Misses
   While studying the Linux *perf* counters for threads and processes we noticed FJ based thread parallelism to incur high amounts of TLB misses. This essentially reduced the number of operations performed per clock cycle making it less efficient than the process model.

In a recent version of DA-MDS we have verified these two effects and have provided a long running thread (LRT) implementation over FJ. Also, by adhering to strict thread process and placement we have reduced the overheads associated with threads. The difference between LRT and FJ implementations is shown in Figure 3-1.

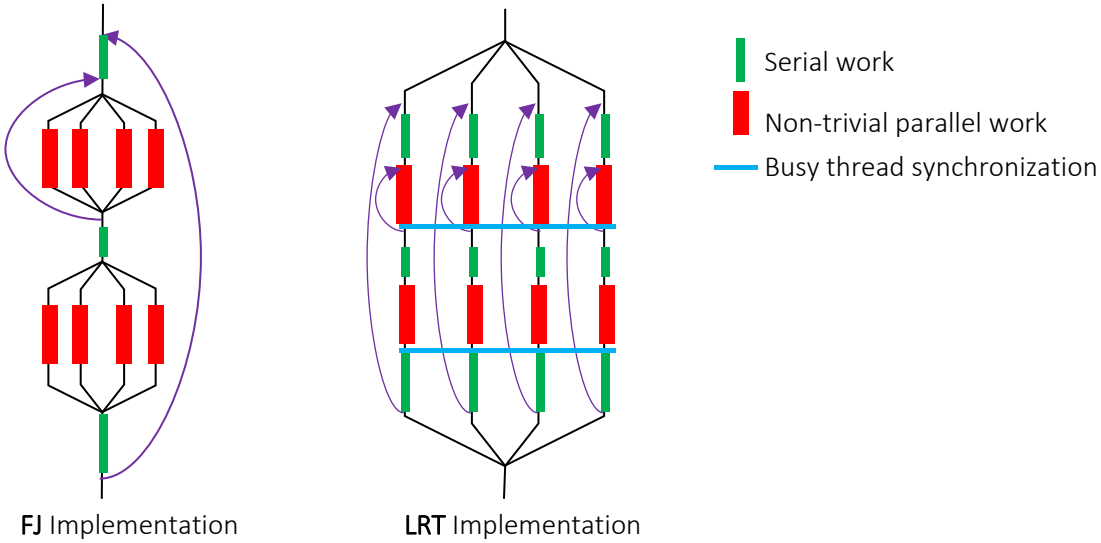


Figure 3-1 Fork-Join FJ vs Long Running Thread LRT implementations

Note, LRT requires a significant amount of code change from what typical MPI+X model programs look like. Also, the programmer is responsible for implementing communication after non-trivial parallel segments where as in FJ the built-in constructs such as *parallel for* implement such synchronization. Moreover, the synchronization implementation needs to make sure that none of the threads "sleep", that is threads would be busy-waiting rather than giving up CPU resources. Performing a Linux *perf* counter analysis we find this produced a smaller number of TLB misses compared to FJ.

Even with LRT implementation, the thread and process placement has to be explicit and within NUMA boundaries to get the best performance. For example, on these 2 socket nodes, placing 1 process with 24 threads is less efficient than placing 2 processes with 1 on each node having 12 threads. It is also important to pin threads to a core. In Java, pinning threads to a core is achieved using OpenHFT's thread affinity library [30].

The point of this experiment with threads was to show that it is possible to achieve similar performance as processes (when process communication is through shared memory), but doing so is not straightforward and requires a considerable amount of code change.

Apart from the intra-node communication optimization, SPIDAL Java employs several other techniques to reduce costs such as Java Garbage Collection (GC), cache and memory access, and heap allocated objects. GC invocations are the so called "stop the world" events, which require all activities within the user code to be stopped while cleaning the heap. These are expensive, especially with these Map-Collective applications where such GC events are responsible for the

strangler effect. SPIDAL Java utilizes off-heap data structures and static allocations to keep GC activity nearly at zero. Also, this makes it possible to run with a minimum memory footprint.

Cache and memory accesses are also important to be optimized in yielding high-performance. SPIDAL Java adopts some of the techniques from scientific simulations such as blocked loops, loop ordering, and 1D arrays to overcome these. It is important to note that data representations with nested data structures add a substantial overhead due to multiple indirect memory references, hence the use of 1D arrays are preferred when possible.

Heap allocated objects require creating temporary copies when used with native I/O operations. Therefore, SPIDAL Java utilizes off-heap memory maps to store such content. This approach is also used in loading initial large data. Memory maps not only offer off-heap allocations, but are significantly faster than the typical Java stream APIs when reading such large data. Also, for inter-node MPI communications these memory maps are more efficient than using heap allocated arrays or objects.

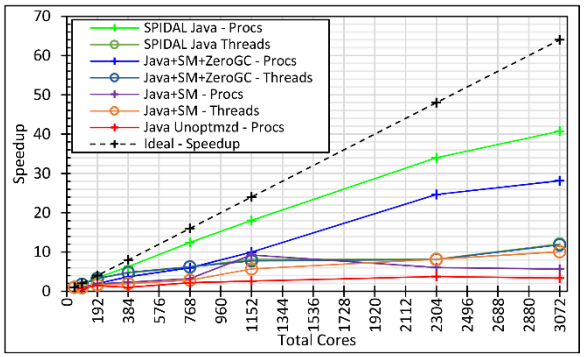


Figure 3-2 DA-MDS speedup for 200K with different optimization techniques

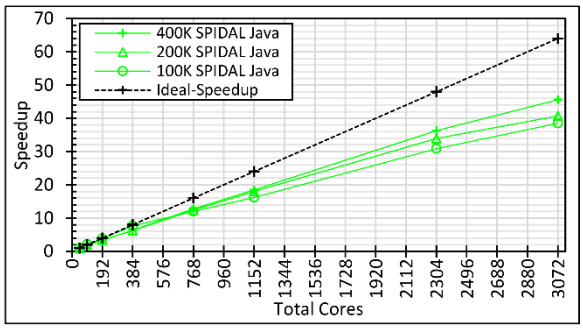


Figure 3-3 DA-MDS speedup with varying data sizes

Figure 3-2 shows the effect of each optimization for DA-MDS as a speedup chart. The results are taken for all processes case. The base case is 48 processes run as 1x1x48, meaning 1 thread per process and 1 process per node across 48 nodes. It shows SPIDAL Java achieves around 40x speedup over 64x core count increase, while typical MPI is only able to achieve 6x speedup for the same increase in cores.

Figure 3-3 shows speedup for varying core counts for three data sizes - 100K, 200K, and 400K. These too were run as all processes because threads did not result in good performance (the tested DA-MDS did not have the LRT implementation discussed above). None of the three data sizes were small enough to have a serial base case, so the graphs use the 48 core as the base, which was run as 1x1x48. SPIDAL Java computations grow(NN^2) while communications grow (NN),

which intuitively suggests larger data sizes should yield better speedup than smaller ones and the results confirm this behavior.

In conclusion, performance results of SPIDAL Java show it scales and performs well in large PC clusters. Also, the optimizations to overcome performance challenges made it possible to run SPIDAL Java applications on much larger data sets than what was possible in the past while still achieving excellent scaling results. The improved shared memory intra-node communication is - pivotal to the gains in SPIDAL Java and it is the first such implementation for Java to the best of our knowledge.

3.3. Cloudmesh Interoperability IaaS and Paas Tool leveraging DevOps

Motivation: Today's cyberinfrastructure is complex and ever changing. Scientist often struggle over the question of how to develop and use next generation big data tools and frameworks. Deployment and use of such infrastructure is complex and often beyond the expertise of data scientists. Furthermore, we have seen scientists perform unnecessary differentiations while using different IaaS platforms such as Openstack, Azure, and AWS. We also identified that the model of generating a virtual machine and using it for a long period of time is broken as security updates and other rapidly developing software render such virtual images obsolete, insecure, and outdated quickly. We need tools and frameworks that makes this easier and allow the creation and recreation of state of the art tools and services used by the data scientists.

Model: Our model targets four layers in the scientific data workflow:

Phase A: IaaS deployment: Creation of virtual clusters that uses an existing HPC or IaaS system

Phase B: PaaS deployment: The platform level in which a platform is deployed or used as part

Phase C: Application deployment: The application deployment and development on A) and B)

Phase D: Data deployment and application execution: The execution of data analysis and experiments while using the programs developed as part of C)

This is achieved while leveraging existing advanced cyberinfrastructure tools and exposing them through a uniform interface.

3.3.1. Virtual Cluster IaaS deployment

We identified that one of the recurring tasks for data scientists is to set up a virtual cluster containing the software needed to perform the actual activities. We have identified through practical experience with data science students that the creation of such clusters often includes sophisticated services that are beyond the capabilities of the scientist to deploy. Furthermore, subtle differences between IaaS frameworks do not allow the generality needed to the experiment on other IaaS offerings and estimate usage impact. Hence we have developed a tool called Cloudmesh that abstracts the IaaS platform and allows easy creation of virtual clusters including proper key management that often is ignored or wrongly executed by the data scientists, who may lack experience in cyberinfrastructure security. An example in Figure 3-8 illustrates the convenience of our tool. Here we demonstrate the use of persistent variables that are integrated in our Cloudmesh command line tool called cm. We can switch with a single variable between clouds, boot, and assign IP addresses and even ssh into the VMs without needing to know all the details about the cloud. An easy configuration simplifies integration of new clouds.

cm default cloud=chameleon cm vm boot cm vm ip assign cm vm ssh	cm default cloud=kilo cm vm boot cm vm ip assign cm vm ssh

Figure 3-8: Booting a VM is simple in Cloudmesh and uniform

While the above also allows the creation of multiple VMs the creation of a virtual cluster requires proper key management between the VMs. This is achieved through our prototype cluster command as illustrated in Figure 3-9 where we boot up 30 virtual machines and allow login between them. In addition, we implemented an inventory command that produces the necessary inventory file used, for example, by Ansible, which is part of Phase B.

cm default cloud=chameleon cm cluster create myCluster –count=30 cm cluster ip assign # not yet implemented cm cluster setup key cm cluster inventory	

Figure 3-9: Booting a cluster of VMs is simple in Cloudmesh

Status: We have developed a prototype of Cloudmesh [31]. It includes an abstraction for OpenStack, and Comet Cloud [1]. Interfaces for AWS, and Azure have been prototyped and demonstrated, but improvements need to be made to integrate it in the production release. The

cluster command has been prototyped but not yet released. The interface to SDSC Comet is still in production. A tutorial will be given at XSEDE2016. A Docker interface is also under development. A prototype to integrate virtualbox vms has been developed. We currently focus on Comet and NSF resources that use OpenStack.

Results: Managing VMs on different IaaS clouds is easy with Cloudmesh. Integration of additional clouds is possible via abstractions. The use of a ***saved state*** in the Cloudmesh client is a distinguishing factor from other efforts. This allows the use of defaults to simplify access to different clouds. We demonstrated use of the following clouds with Cloudmesh: FutureSystems, Chameleon Cloud, Jetstream, CloudLab, Cybera, AWS, Azure, VirtualBox

3.3.2. Virtual cluster PaaS, Data and Application deployment

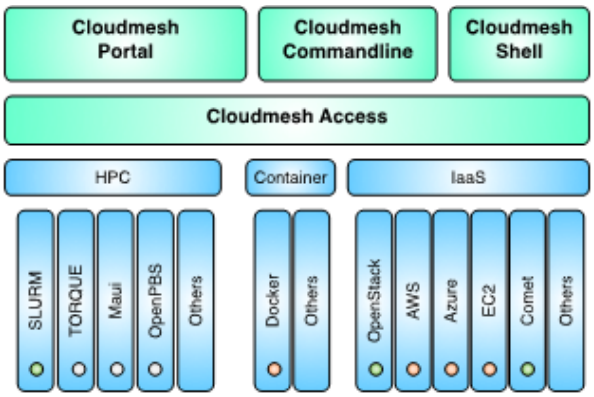


Figure 3-10: Architecture of the Cloudmesh abstraction layers to gain access to cyberinfrastructure systems. DevOps frameworks are available as part of the Cloudmesh access to them and are coordinated and choreographed with the help of the shell, command line or a portal interface that we will develop.

Once a virtual cluster either as HPC, VMs, or containers is available additional software services need to be installed on such a system. This can be achieved while leveraging software configuration tools in support of DevOps such as Ansible, Chef, Pupet, Saltstack or others. In our efforts we have focused so far on ansible as the deployment framework as it allows us to leverage a deployment methodology based on well-known security concepts and abstractions allowing a push model. Just as we can deploy such platforms we are currently evaluating to use the same deployment frameworks for application data, software and even their execution.

Status: We have developed prototype deployments for several Apache based tools and services such as Hadoop and Spark. We have tested them on Openstack within Futuresystems and Chameleon cloud. Last semester we supported and evaluated the use of the framework and its tools in a "Big Data Open Source Software Projects Class" that had 40 teams with various projects in big data deployment. Based on the experience of the class we have identified that using Cloudmesh cluster will introduce much more flexibility and ease of use for the data scientists. Furthermore, we can introduce an additional abstraction layer that would allow us to integrate multiple deployment frameworks and not just

focus on a single DevOps tool such as ansible. We have made initial good progress while also using the DevOps frameworks for the application data and software deployment.

The access to the sophisticated cyberinfrastructure is summarized in Figure 3-10.

Summary of Cloudmesh

- Cloudmesh was downloaded 287 times in April (However since then pypi has discontinued their download information so we have no further information on downloads).
- We are presenting a tutorial at XSEDE2016 that uses Cloudmesh
- We have written a paper that was accepted at XSEDE2016 using Cloudmesh [32]
- We have identified that Cloudmesh significantly reduces startup time and effort to use multiple IaaS
- We are using the Cloudmesh principles in the current summer REU activity.
- Cloudmesh is now used to support the open science virtual cluster on Comet.

3.4. Pilot Jobs and Pilot Data Memory

Motivation: The Pilot-Abstraction offers a unified approach for application-level compute and data management across heterogeneous compute resources (e.g. HPC, cloud, Hadoop), storage resources (e.g., local disks, cloud storage, parallel filesystems, SSD) and memory. As part of MIDAS we extended the Pilot-Abstraction to facilitate the integration of ABDS and HPC at the level of scheduling (Yarn, Slurm) and data access integrating ABDS HDFS, in-memory systems (Spark) and HPC file systems (Lustre).

With the introduction of YARN a broader set of applications can be executed within Hadoop clusters than ever before. However, developing and deploying YARN applications potentially side-by-side with HPC applications remains a difficult task. We still lack established abstractions that are easy-to-use while still enabling the user to reason about compute and data resources across infrastructure types (i.e., Hadoop, HPC and clouds).

YARN provides a low-level abstraction for resource management, e.g., a Java API and protocol buffer specification. Typically interactions between YARN and the applications are much more complex than the interactions between an application and a HPC scheduler. Further, applications must be able to run on a dynamic set of resources; YARN can preempt containers in high-load

situations. Data/compute locality need to be manually managed by the application scheduler by requesting resources at the location of a file chunk. Also, allocated resources (the so called YARN containers) can be preempted by the scheduler.

To address these shortcomings, various frameworks that aid the development of YARN applications have been proposed [33]. While these frameworks simplify development, they do not address concerns such as interoperability and integration of HPC/Hadoop. To facilitate the uptake of Hadoop ecosystem in an HPC context we integrate YARN and SPARK into the RADICAL-Pilot framework, so as to provides advanced and scalable data analysis capabilities to existing high performance applications while allowing applications to run HPC and Hadoop application parts side-by-side. These implementations are called **Pilot-YARN and Pilot-SPARK**

We extended RADICAL-Pilot, to support the deployment and management of the Hadoop/Spark cluster to the resources acquired. The extension of RADICAL-Pilot was mainly done to the RADICAL-Pilot's Agent. The Agent has the following components: The Agent Execution Component, the Heartbeat Monitor, Agent Update Monitor, Stage In and Stage Out workers. The integration of Hadoop/Spark was done in the agent's execution component. Figure 3-11 shows how YARN/Spark specifics were integrated in the RADICAL-Pilot Agent.

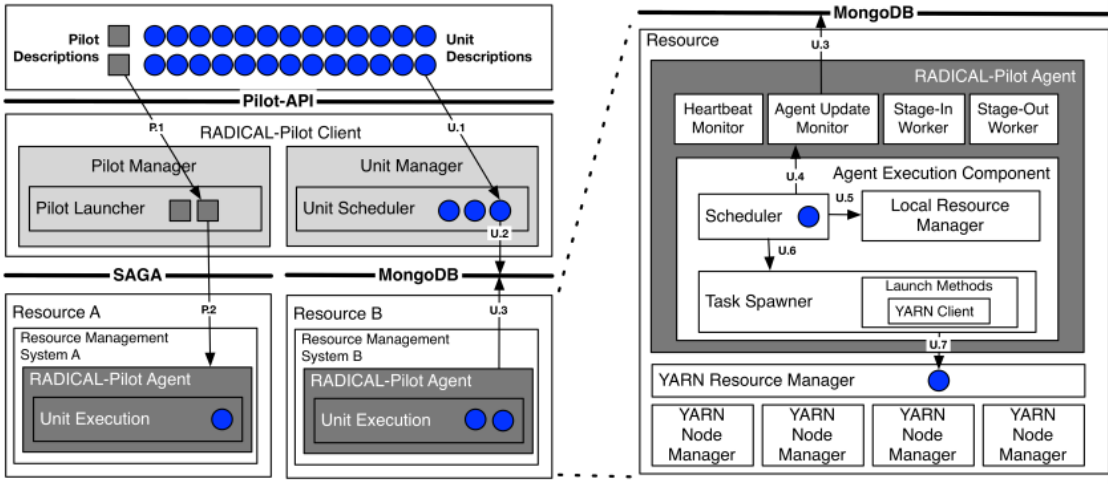


Figure 3-11: RADICAL-Pilot YARN Architecture. All YARN/Spark specifics are encapsulated in the RADICAL-Pilot Agent

Pilot-Data and Pilot-Memory: Pilot-Data [34] offers a unified approach for data management in conjunction with Pilot-Jobs across complex storage hierarchies comprised of local disks, cloud storage, parallel filesystems, SSD and memory. Doing so allows the efficient management of Pilot-/task-level input data as well as intermediate and output data taking into account data locality.

While this disk-based model is suitable for compute-bound tasks, for scalable data processing – like data transformations using the split-apply-combine pattern, more sophisticated methods are required. The usage of memory allows the efficient caching of input and intermediate data, which it essential for these algorithms.

We propose Pilot-Data Memory as both an extension to Pilot-Data and as a runtime system for supporting an increasing number of iterative algorithms. Pilot-Data Memory supports application patterns, such as the split-apply-combine pattern and iterative algorithms, as well as KMeans or optimizations algorithms. It adds in-memory capabilities to Pilot-Data and makes it available via the Pilot-API. Figure 3-12 shows the architecture of Pilot-Data Memory.

An important design objective for Pilot-Data Memory is extensibility and flexibility. Pilot-Data Memory supports different in-memory backends: (i) file-based, (ii) in-memory Redis and (iii) in-memory Spark. Pilot-YARN and Pilot-Spark can be used to setup the necessary Spark infrastructure on a HPC resource.

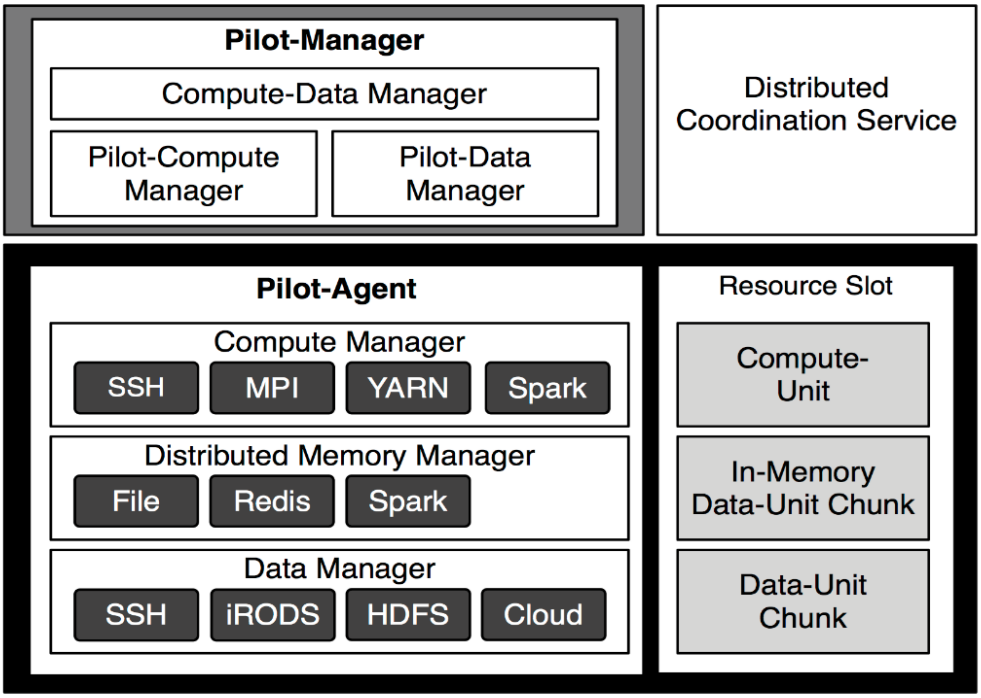


Figure 3-12 Pilot Data and Pilot-Data Memory Architecture

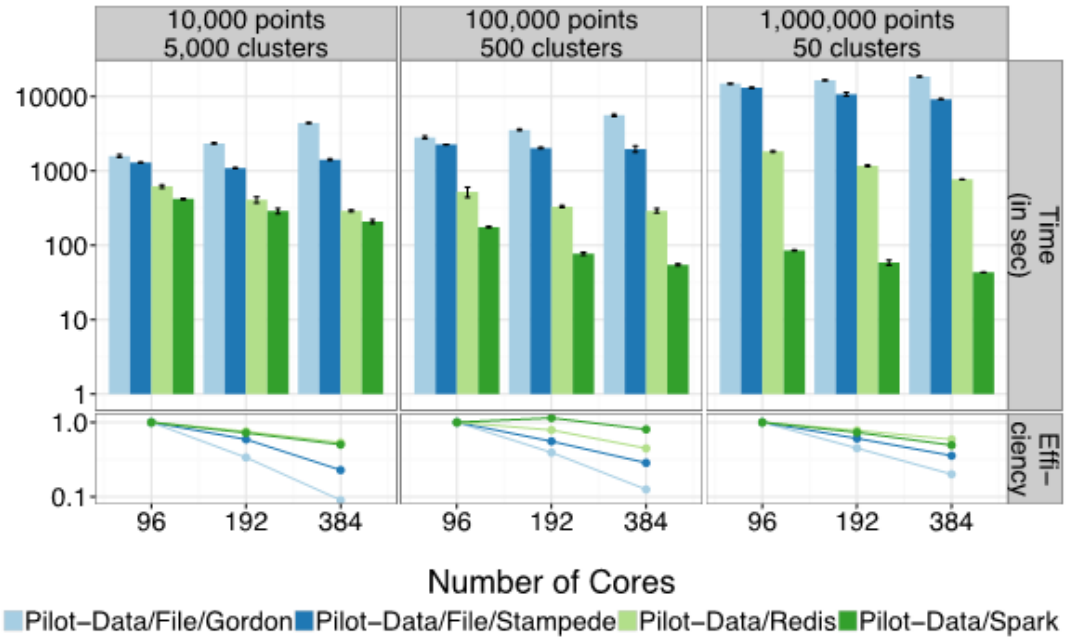


Figure 3-13: KMeans Pilot-Data: Running KMeans on Different Pilot-Data Backends. The iterative KMeans algorithms benefits from the usage of Spark and Redis In-Memory Backends.

The different backends are supported via an adaptor service interface that specifies the capabilities necessary for implementation by the in-memory backend; it consists of functions for allocating/de-allocating memory, loading data and executing map/reduce functions on the data. Depending on the backend the processing function must be implemented either manually, e.g., file-based and Redis backend adaptors, or directly delegated to the processing engine like for Spark. The Redis and file backends use the Pilot-Job framework for executing the Complete Units generated by Pilot-Data Memory. If required, the application can access the native runtime functions via a context interface. It is important to note that Pilot-Data Memory can be easily extended to other backends, e. g. Alluxio and HDFS in-memory storage tier which we will evaluate. Performance measurements are shown in Figure 3-13.

3.5. Architecture of Scalable Big Data Machine Learning Library

Motivation: This section establishes principles for designing parallel machine learning algorithms supporting a variety of model synchronization paradigms. It suggests a bridge between parameter server approaches and those "owner compute rule-based" distributed model parameter approaches familiar in HPC.

There is a vast amount of literature on distributed machine learning and data analytics, much of it continuing a long tradition of developing special ways to speed up or parallelize individual algorithms or applications. However, specialized implementation rarely leads to wide-spread deployment since it yields no generalization of parallelization techniques. Thus the focus of our work [19] is to develop a general and exact parallelization technique for a large class of machine learning algorithms. It aims to provide the software building blocks (kernels) that are portable to manycore (and GPU) architectures, as we migrate from the multicore to manycore era.

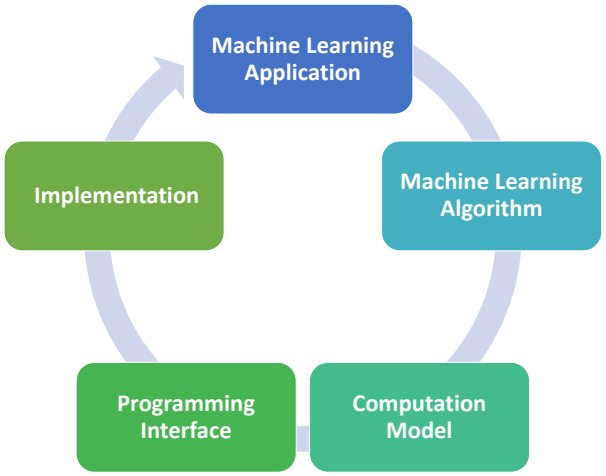


Figure 3-14. A solution for big data machine learning application includes decisions on algorithms, computation models, programming interfaces, and implementation.

We define the process for parallelization of machine learning algorithms as shown in Figure 3-14: the first step is to choose an algorithm for a given big data analysis problem. It may occur that there are multiple solutions to the same problem. An implementation is often optimized for a selected algorithm. Such a tightly coupled cycle (ref. top triangle of Figure 3-14) works well for a specific application b but becomes difficulty to sustain due to diverse choices as well as changes of technology at algorithm, system and hardware levels. This motivates us to investigate the fundamental issue of **computation and parallelization abstractions** that are effective for a set of domain problems.

We propose a systematic approach with categorizations based on "**Computation Model**", which effectively expresses kernel computation characteristics and synchronization or communication mechanisms. The separation of Computation Model, Abstraction and Implementation details allows us to adapt the variants and make the optimization easier for parallel and distributed machine learning algorithms. Programming interface in particular provides APIs to application users.

- **Computation Model**
  High level description of the parallel algorithm, not associating with any execution environment.

- **Abstraction**
  Middle level description of the parallelization, associating with a programming framework or

runtime environment and including the data abstraction/distribution, processes/threads and the operations/APIs for performing the parallelization (e.g. network and manycore/GPU devices).

- **Implementation**
  Low level details of implementation (e.g. language).

We further categorize parallel machine learning applications into four types of computation models (see Figure 3-15):

Computation Model A

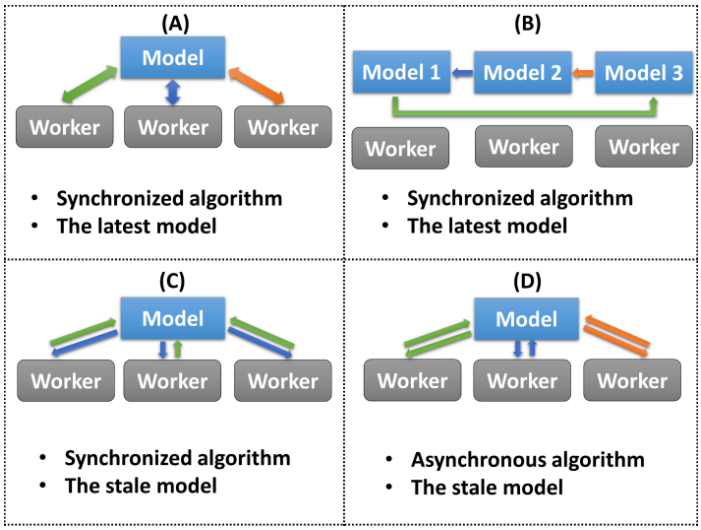


Figure 3-15. Four Computation Models

This computation model uses a synchronized algorithm to coordinate parallel workers. In each iteration, once a worker processes a training data item, it locks related model parameters and prevents other workers from accessing them. When the related model parameters are updated, the worker unlocks the parameters. As long as workers compute and update on different model parameters, they can execute in parallel. Only one worker is allowed to access a word's model parameters at a time; therefore the model parameters used in the local computation are always the latest. In practice, this computation model is seldom applied due to the high overhead of locking.

Computation Model B

The next computation model also uses a synchronized algorithm. Each worker first takes a partition of the shared model and performs computation. Afterwards, the model partitions are shifted between workers. When all the model partitions are accessed by all the workers, an iteration is complete. Through model rotation, each model parameter is computed and updated by only one worker at a time so that the consistency of the model is maintained.

Computation Model C

Computation Model C applies a synchronized algorithm but with a stale model. In a single iteration,

each worker first fetches all the model parameters required by local computation. When the local computation is completed, the modifications of the local model from all the workers are combined to update the model.

Computation Model D
With this model, an asynchronous algorithm employs a stale model. Each worker independently fetches the related model parameters, performs the local computation, and returns the model updates. Unlike Computation Model A, other workers are allowed to fetch or update the same model parameter independently. In contrast to Computation Model B and C, there is no synchronization barrier in this computation model.

Based on the summarized computation models, we propose a new set of model-centric abstractions including data abstraction and synchronization operation abstraction for parallel machine learning applications as a part of the MapCollective model. These establish parallel machine learning as the combination of training data-centric and model-centric processing. The new model-centric computation abstractions can support numerous, including but not limited to:

- Expectation-Maximization Type
    - K-Means Clustering
    - Collapsed Variational Bayesian for topic modeling (e.g. LDA)
- Gradient Optimization Type
    - Stochastic Gradient Descent and Cyclic Coordinate Descent for classification (e.g. SVM and Logistic Regression), regression (e.g. LASSO), collaborative filtering (e.g. Matrix Factorization)
    - Markov Chain Monte Carlo Type
      Collapsed Gibbs Sampling for topic modeling (e.g. LDA in Section 4.2.)

3.6. Harp Programming Paradigm

Motivation: This section introduces Harp [26], whose basic idea is to abstract iterative communication and scientific data abstractions with *MapCollective*. It distinguishes distributed (training) data and distributed model (parameters) and these set up parallel machine learning as the combination of training data-centric and model-centric parallel processing. Harp distributes

model over worker nodes and supports *collective communication* to bring global model to each worker node.

The Harp Programming Paradigm shown in figure 3-16, abstracts parallel applications in the MapCollective model which is extended from the original MapReduce model. Here parallelization of an application is abstracted as parallel execution on a set of Map tasks which are synchronized with collective communication operations. While the input data is abstracted and partitioned as KeyValue pairs, the abstraction of the synchronized model data and related collective communication operations are specially defined. These ideas are implemented in the Harp library (open source) as a Hadoop plugin. By plugging Harp into Hadoop, the MapCollective model can be expressed on top of a MapReduce framework and efficient data synchronization for a variety of machine learning applications is enabled. In addition, mapping a MapCollective model to Hadoop also enables two levels of parallelism. Since each Map task is a process where the collective communication operations are invoked and multi-thread execution is enabled for another level of parallelism.

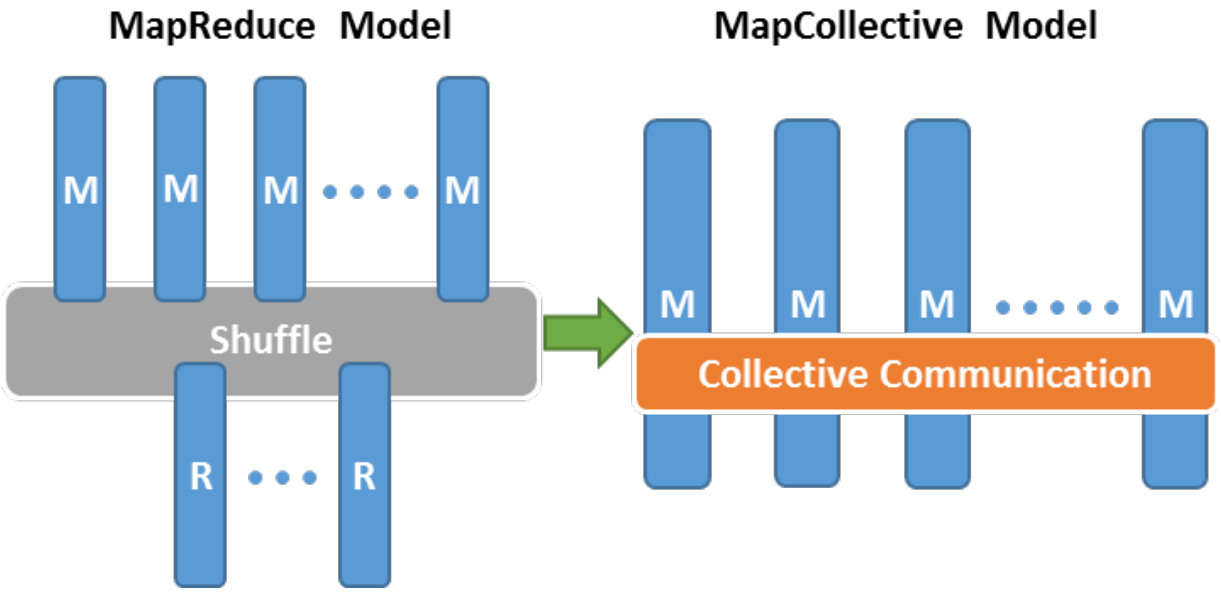


Figure 3-16. MapCollective Model and Hadoop Plugin

The data types in Harp are abstracted in a hierarchy. Data are horizontally abstracted as arrays or key-values and constructed from basic types into partitions and tables vertically. At the lowest level, there are two basic types: arrays and objects. Based on the component type of an array, there can be byte array, int array, long array or double array. Object type is used to describe keys and values. In the middle level, arrays and objects are wrapped as array partition and key-value partition. At the top level are tables containing multiple partitions, each with a unique partition ID.

Tables on different parallel workers can be associated with each other and present one dataset. The collective communication operations are defined as redistribution or consolidation of partitions in tables.

Table 3-1: Collective Operations supported in Harp

Operation	Algorithm	Time Complexity
broadcast	chain	$n\beta$
	minimum spanning tree	$(\log_2 p)n\beta$
reduce	minimum spanning tree	$(\log_2 p)n\beta$
allgather	bucket	$pn\beta$
allreduce	bi-directional exchange	$(\log_2 p)n\beta$
	regroup-allgather	$2n\beta$
regroup	point-to-point direct sending	$n\beta$
send messages to vertices	point-to-point direct sending	$n\beta$
syncLocalWithGlobal	point-to-point direct sending plus routing optimization	$pn\beta$
syncGlobalWithLocal	point-to-point direct sending plus routing optimization	$n\beta$
rotateGlobal	direct sending between neighbors	$n\beta$

Note in Column "Time Complexity", p is the number of processes, n is the number of input data items per worker, β is the per data item transmission time, communication startup time α is neglected and the time complexity of the "point-to-point direct sending" algorithm is estimated regardless of potential network conflicts.

Collective communication operations are defined on top of the data abstractions. The operations are abstracted based on the synchronization mechanisms summarized from the existing tools and many applications for learning. Currently four categories of collective communication operations are supported: (1) operations adapted from MPI: e.g. "broadcast", "reduce", "allgather", and "allreduce"; (2) operations derived from MapReduce: e.g. "regroup" operation with "combine & reduce" support; (3) operations derived from graph processing tools: e.g. "send messages to vertices"; and (4) operations abstracted from machine learning applications with big models: e.g. "syncLocalWithGlobal" and "syncGlobalWithLocal", or "rotate".

The collective communication operations are not specific to some data abstractions. For each operation, both arrays and objects can be used. Even for graph-based communication, the operations are not tied with graph abstractions. Instead, the data movement is between partitions according to their locality. For each operation, routing mechanisms are optimized based on the total data size involved in the movement. Routing optimization is very helpful to many iterative applications in which synchronization happens in iterations, especially to machine learning applications which need to frequently synchronize the huge model. We apply this MapCollective model to Latent Dirichlet Allocation (LDA) and show that with MapCollective abstractions the implementation can achieve better performance compared with parameter server type applications which use asynchronous communication methods.

3.7. Integration of Harp and Intel DAAL Library

Motivation: For machine learning libraries and it is obviously advantageous to reuse highly optimized kernels as software building blocks. Intel's Data Analytics Acceleration Library (DAAL) [43] provides several core algorithms with excellent intra-node parallelism. Here we explore using Harp to invoke DAAL and thus build a distributed version of this library.

We aim to combine the advantages of Intel's DAAL for intra-node multithreading and Harp programming framework for inter node communications. Intel optimizes a select group of data analytics and machine learning algorithm kernels on their hardware platforms, from CPUs to more recent Xeon Phi coprocessor of manycore architectures. As an extension to its highly reputable Math Kernel Library (MKL), DAAL provides high performance on its batch mode. Yet the performance of its kernels on distributed mode relies on the communication framework chosen by the users, which motivates our effort to interface Harp with DAAL. Harp is designed to handle communication overheads within iterative applications by using collective in-memory communication operations. However, the implementation of local computation within the current version of Harp, which is written in multithreading Java, is not straightforward in memory management. Thus, an integrated Harp-DAAL programming framework shall result in a significant improvement of the performance.

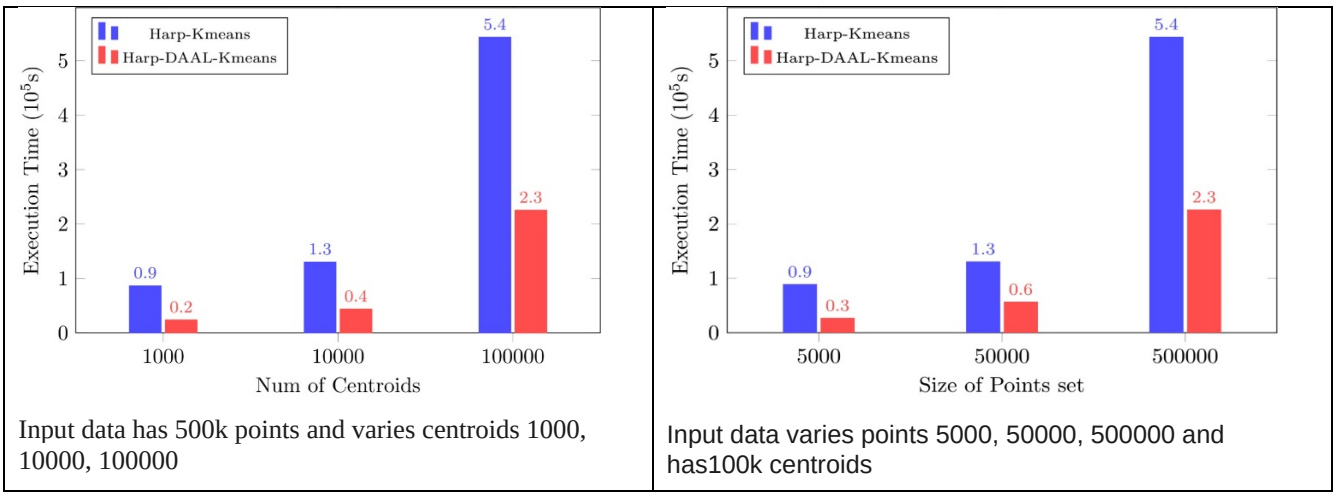


Figure 3-17. K-means Performance of Harp and Harp-DAAL

Fig. 3-17 shows a performance comparison between two implementations of K-means clustering. The experiments are done on two nodes of Haswell Xeon processor within a cluster of the FutureSystems. The K-means kernel with local computation offloaded to DAAL (red lines) achieves significantly lower execution time than the kernel implemented by Java threads.

The Harp-LDA algorithm is implemented using Java threads, while the Harp-DAAL-LDA algorithm takes advantage of the optimized native computation kernels on Intel's platform. Unlike K-means clustering, LDA has complicated irregular memory access, which requires more effort to reduce the memory data transfer overheads. Moreover, there is no LDA kernel within the current version of DAAL. We need to write the native LDA while calling the optimized MKL kernel at low levels. Our approach includes two aspects:

Data type conversion between Harp and native kernels

Harp, due to its optimization on collective communication among nodes, adopts a massive use of memory allocation in a nonconsecutive way. In contrast, DAAL and MKL allocate the data on contiguous memory chunks, which better fits the requirement of data alignment within BLAS operations. A compromise should be made between the two aspects and so we attempt to create some highly efficient data conversion methods. In order to profile the memory usage of kernels within the Harp-DAAL framework, we use Intel VTune Amplifier as a profiling tool. We also conceive a way to profile the Harp/Hadoop applications by VTune, though VTune is mainly used for profiling programs written in native languages.

Memory Optimization on Intel's Xeon Phi Knights Landing

According to the announcement of Intel's latest 2nd generation of Xeon Phi coprocessor, codenamed Knights Landing (KNL), we intend to optimize the Harp-DAAL-LDA kernel especially on that platform. KNL has tremendously improved its memory latency and bandwidth compared to the previous generation. The so-called multi-channel dynamic random access memory (MCDRAM) has on-package high bandwidth memory technology (HBM), which could either serve as a last level of cache between the L2 cache and the on-platform DDR4 memory or as a separate memory node alongside the DDR4 memory. According to [44] MCDRAM can significantly accelerate the latency-bounded applications that are usually hard to achieve on Xeon and the first generation of Xeon Phi. Therefore, we will leverage the potential benefits of KNL on Harp-DAAL-LDA applications.

4 SPIDAL Algorithms

4.1. Introduction

4.2. Harp Latent Dirichlet Allocation

4.3. SPIDAL Algorithms – Subgraph mining

4.4. SPIDAL Algorithms – Random Graph Generation

4.5. SPIDAL Algorithms – Triangle Counting

4.6. SPIDAL Algorithm – Community Detection

4.7. SPIDAL Algorithms – Core

4.8. SPIDAL Algorithms – Optimization

4.9. SPIDAL - Algorithms Polar Remote Sensing Algorithms

4.10. SPIDAL Algorithms – Nuclei Segmentation for Pathology Images

4.11. SPIDAL Algorithms – Spatial Querying Methods

4.1. Introduction

In the original proposal, we identified a set of algorithms to address in SPIDAL

Table 4-1 Status & Parallelism Abbreviations Used in Tables 4-2 to 4-4

GML	Global (parallel) Machine Learning	ToDo	No prototype Available
PP	Pleasingly Parallel (Local ML)	Seq	Sequential version Available
GrA	Good distributed algorithm needed	P-DM	Distributed memory parallel algorithm Available
GrB	Graphs with runtime parallel partitioning	P-ShM	Shared memory parallel algorithm Available
GrC	Graphs with static parallel partitioning		

Table 4-2 Proposed SPIDAL Algorithms for Graphs and Spatial Analytics

Algorithm	Applications	Features	Status	Parallelism
Graph Analytics				
Community Detection	Social Networks, webgraph	Graph	P-DM	GML-GrC
Subgraph/motif finding	Webgraph, biological/social networks		P-DM	GML-GrB
Finding diameter	Social networks, webgraph		P-DM	GML-GrB
Clustering coefficient	Social networks		P-DM	GML-GrC
Page rank	Webgraph		P-DM	GML-GrC
Maximal cliques	Social networks, webgraph		P-DM	GML-GrB
Connected component	Social networks, webgraph		P-DM	GML-GrB
Betweenness centrality	Social networks	Graph, Non-metric, static	P-Shm	GML-GRA
Shortest Path	Social networks, webgraph		P-Shm	
Spatial Queries and Analytics				
Spatial relationship based queries	GIS/social networks/pathology/ informatics	Geometric	P-DM	PP
Distance based queries			P-DM	PP
Spatial clustering			Seq	GML
Spatial modeling			Seq	PP

Table 4-3 Proposed SPIDAL Algorithms for Image Processing and Deep Learning

Algorithm	Applications	Features	Status	Parallelism
Core Image Processing				
Image preprocessing	Computer vision/pathology informatics	Metric Space Point sets, Neighborhood sets & Image features	P-DM	PP
Object detection & segmentation			P-DM	PP
Image/object feature computation			P-DM	PP
3D image registration			Seq	PP
Object matching		Geometric	Todo	PP
3D feature extraction			Todo	PP
Deep Learning				
Learning Network, Stochastic Gradient Descent	Image Understanding, Language Translation, Voice Recognition, Car driving	Connections in artificial neutral net	P-DM	GML

Table 4-4 Proposed SPIDAL Core and Optimization Algorithms

Algorithm	Applications	Features	Status	Parallelism
DA Vector Clustering	Accurate Clusters	Vectors	P-DM	GML
DA Non-metric Clustering	Accurate Clusters, Biology, Web	Non metric, $O(N^2)$	P-DM	GML
Kmeans; Bsic, Fuzzy and Elkan	Fast Clustering	Vectors	P-DM	GML
Levenberg-Marquardt Optimization	Non-linear Gauss Newton, use in MDS	Least Squares	P-DM	GML
SMACOF Dimension Reduction	DA-MDS with general weights	Least Squares, $O(N^2)$	P-DM	GML
Vector Dimension Reduction	DA-GTM and others	Vectors	P-DM	GML
TFIDF Search	Fine nearest neighbors in document corpus	Bag of "words" (image features)	P-DM	PP
All-pairs similarity search	Find pairs of documents with TFIDF distance below a threshold	Bag of "words" (image features)	Todo	GML
Support Vector Machine SVM	Learn and Classify	Vectors	Seq	GML
Random Forest	Learn and Classify	Vectors	P-DM	PP
Gibbs sampling (MCMC)	Solve global interference problems	Graph	Todo	GML
Latent Dirichlet Allocation LDA with Gibbs sampling or Var. Bayes	Topic models (Latent factors)	Bag of "words"	P-DM	GML
Singular Value Decomposition SVD	Dimension Reduction and PCA	Vectors	Seq	GML
Hidden Markov Models (HMM)	Global inference on sequence models	Vectors	Seq	PP & GML

In addition, there are community specific analytics often building on some of those in Tables 4-2 to 4-4. Table 4-2 is covered in subsections 4.3, 4.4 and 4.5. Table 4-3 is covered in subsections 4.8 to 4.10, while Table 4-4 is covered in subsections 4.2, 4.6 and 4.7.

4.2. SPIDAL Algorithms – Harp Latent Dirichlet Allocation

Motivation: Latent Dirichlet Allocation is an important algorithm that is representative of several related sophisticated latent factor (topic) determination problems. Further it involves data structures and can benefit from loosening synchronization between model parameters in the different processes of a parallel algorithm. It was thus a natural case to investigate with the Harp MIDAS technology which had been proven in simpler cases – especially DA-MDS reported later under core machine learning.

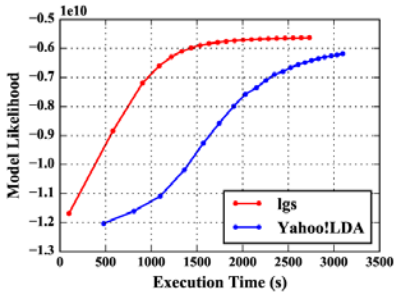
The research work focuses on the computation models and the synchronization mechanisms of parallel machine learning applications using Latent Dirichlet Allocation (LDA) as an example [25]. LDA is a widely used machine learning technique for big data analysis, including text mining, advertising, recommender systems, network analysis, and genetics. We use Collapsed Gibbs Sampling (CGS) algorithm to solve LDA. A major challenge is the scaling issue in parallelization owing to the fact that the model size is huge and parallel workers need to synchronize the model continually. We identify three important features of the model in parallel LDA CGS computation: (1) the model volume required for local computation is high; (2) the time complexity of local computation is proportional to the related model size; (3) the model size shrinks as it converges. By investigating collective and asynchronous methods of the model synchronization mechanisms, we discover that optimized collective communication can improve the model update speed, thus allowing the model to converge faster. The performance improvement derives not only from accelerated communication but also from reduced iteration computation time as the model size shrinks during the model convergence. To foster faster model convergence, we design new collective communication abstractions and implement two Harp-LDA applications, “lgs” and “rtt”.

We compare our new approach with Yahoo! LDA and Petuum LDA, two leading implementations favoring asynchronous methods in the field, on a 100-node, 4000-thread Intel Haswell cluster with three different datasets (see Table 4-5). When using local-global model synchronization on “enwiki”, “lgs” reaches higher model likelihood with shorter execution time (see Fig. 4-1a). Though “lgs” can be overtaken by Yahoo! LDA on “clueweb”, by increasing model synchronization rounds per iteration to four, “lgs-4s” obtains higher model convergence speed (see Fig. 4-1b). When using model rotation, “rtt” and Petuum LDA achieve similar model likelihood with similar execution time on “clueweb” (see Fig. 4-1c). However, on “bi-gram”, as the number of words in the model grows,

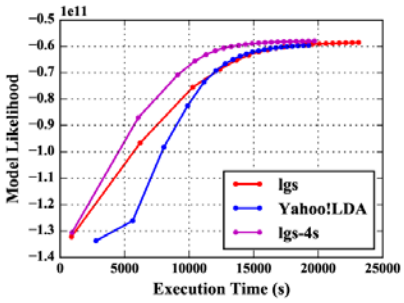
"rtt" runs 3.9 times faster compared with Petuum LDA (see Fig. 4-1d). The details of this research work are described in [14].

Table 4-5 Training Data Settings

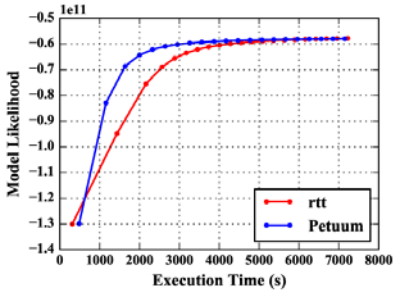
Dataset	Number of Docs	Number of Tokens	Vocabulary	Doc Length Mean/SD	Number of Topics	Initial Model Size
clueweb	50.5M	12.4B	1M	224/352	10K	14.7GB
enwiki	3.8M	1.1B	1M	293/523	10K	2.0GB
bi-gram	3.9M	1.7B	20M	434/776	500	5.9GB



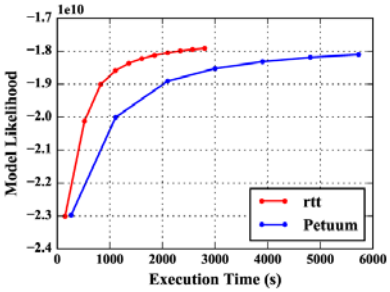
(a) Execution Time vs. Model Likelihood on "enwiki"



(b) Model Likelihood vs. Execution Time on "clueweb"



(c) Model Likelihood vs. Execution Time on "clueweb"



(d) Model Likelihood vs. Execution Time on "bi-gram"

Figure 4-1. Performance comparison between "lgs" and Yahoo! LDA and Performance comparison between "rtt" and Petuum

4.3. SPIDAL Algorithms – Subgraph mining

Motivation: Subgraph isomorphism is a canonical problem in several where people are interested in finding subsets of nodes with specific labels or attributes and mutual relationships that match a specific template, such as social network analysis [35], data mining [36, 37], fraud detection [38] and bioinformatics [39]. For example, in financial networks, where the nodes are

banks/individuals, and edges represent financial transactions, an investigator might be interested in specific transaction patterns from an individual to banks, e.g., through suspicious intermediaries to deflect attention [38]. In many bioinformatics applications, frequent subgraphs (referred to as "motifs") in protein-protein interaction networks (PPI) have been used to characterize the network, distinguish it from random networks and identify functional groups [39, 40].

Relational subgraph analysis, e.g. finding labeled subgraphs in a network, which are isomorphic to a template, is a key problem in many graph related applications. It is computationally challenging for large networks and complex templates, and thus we are working on algorithms for relational subgraph analysis using Harp. We study a variety of subgraph isomorphism problems, such as: (i) counting the number of embeddings of a given labeled/unlabeled template; (ii) finding the most frequent subgraphs/motifs efficiently from a given set of candidate templates; and (iii) computing the graphlet frequency distribution.

By plugging Harp into Hadoop, we can express the MapCollective model in a MapReduce framework and enable efficient in-memory collective communication between map tasks. It stores the intermediate data (or model data) on all nodes, each node with a different partition.

An algorithm for subgraph analysis using Hadoop, called Sahad, is given in [41], which is based on a color-coding scheme [42].

Network	No. Of Nodes (in million)	No. Of Edges (in million)	Size (MB)
Web-google	0.9	4.3	65
Miami	2.1	51.2	740

Table 4-6. Networks of Graph Applications

Figure 4-2. Sub-graph Templates

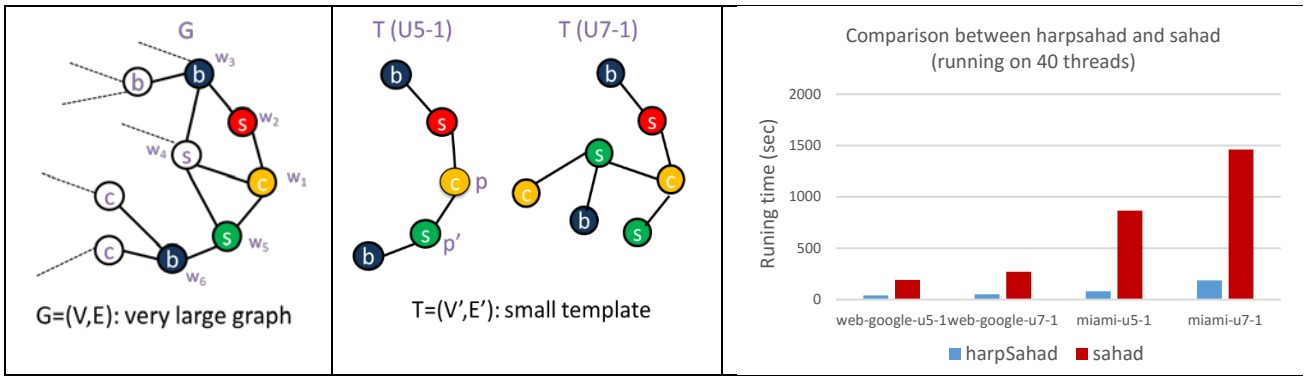


Figure 4-3. Performance comparison between SAHAD and HarpSahad on Web-google and Miami networks using Template u5-1 and u7-1.

We used two networks as shown in Table 4-6 in the experiments. Figure 4-2 shows four unlabeled templates used. Figure 4-3 displays the performance comparison between HarpSahad and Sahad. We ran the experiments on the Juliet cluster at Indiana University using 40 threads (4 nodes and 10 threads per node). Juliet is a cluster with Haswell architecture and has 128 nodes in total. For network Web-google, HarpSahad runs about 5 times faster than Sahad, and for Miami network, HarpSahad performs about 9 times faster than Sahad

We're working on further improvements in the follow areas:

1. **Memory usage optimization.** The initial Sahad implementation cannot work on very large template and networks. This issue is challenging due to the growing model data, as in our case.

2. **Communication overhead.** We are working at communication models for specific partitioning schemes. The rotation model is not suitable for this application because it may transfer unnecessary model data from one node to another, which increases the communication overhead.

3. **Load balancing.** We investigated several partitioning schemes such as random-partition, minimum-cut, and even-partition. We also looked at increasing parallelism on high edge count nodes.

4.4. SPIDAL Algorithms – Random Graph Generation

Motivation: Advances in hardware technology as well as the developments in software and algorithms have enabled the detailed study of complex networks such as the Internet, biological networks, social networks, and various infrastructure networks. The study of these complex systems has significantly increased the interest in various random graph models. Many real-world systems and networks are modeled and analyzed using various random graph models. These models must incorporate relevant properties such as degree distribution and clustering coefficient. Many of them, such as the preferential attachment (PA) model, Chung-Lu (CL), stochastic Kronecker, stochastic block model (SBM), and block two–level Erdos-Renyi (BTER) models, have been devised to capture those properties. As some of the complex networks grow, it has become necessary to correspondingly generate massive random networks efficiently. A smaller network may not exhibit the same behavior, even if both networks are generated using the same model. The generative algorithms for these models are mostly sequential and take a prohibitively long time to create large-scale graphs. We are working on developing efficient parallel algorithms for producing random graphs using various models.

We have developed a novel method (called the DG method), based on grouping the vertices by their degrees, that leads to space- and time-efficient sequential and parallel algorithms for several random graph models, including the CL model, with rigorous guarantees. Our main contributions are summarized below.

1. Space efficiency: Both of our sequential and parallel algorithms for the CL model require only O(∧) space, where ∧ is the number of distinct degrees, comparing to O(n) space required by the previous algorithms. In the real-world networks, ∧ is significantly smaller than n. Experimental results on a wide range of large-scale networks show that our algorithms require 400–15000 times less memory than the previous algorithms. This space efficiency makes our algorithms suitable for generating very large-scale graphs.

2. Time efficiency: Our algorithms are more efficient in terms of runtime also. We prove that our sequential and parallel algorithms have running time of O(m) and O($m/P + \wedge + P$), respectively, with high probability, where P denotes the number of processors. In contrast to earlier algorithms, the associated constants and overheads are significantly smaller for our algorithms. Experimental results show that our algorithms are about 3-4 times faster than the previous algorithms. Moreover, our parallel algorithm achieves almost optimal load balancing using an efficient load balancing technique and scales very well to a large number

of processors. Our parallel algorithm can generate a network with 250 billion edges in just 12 seconds using 1024 processors.

3. Extensions to other models: Finally, we show how our algorithmic method extends naturally to the BTER and SBM models and leads to significantly improved sequential and parallel algorithms. Experimental results show that after applying the DG method, the runtime for the BTER model improves by a factor of 5–80 for various types and sizes of networks.

Figure 4-4 shows the performance of our DG algorithm against the MH algorithm [45], which is the best-known previous sequential algorithm, using both real-world and synthetic networks. We extracted the degree sequences of these networks, and then generated new graphs from these degree sequences. We observe that our DG algorithm is approximately 3 times faster than the MH algorithm as we discussed before. A huge improvement made by our algorithm is on the memory requirement, improving it by a factor of 440–3474 for the networks shown in the Figures below.

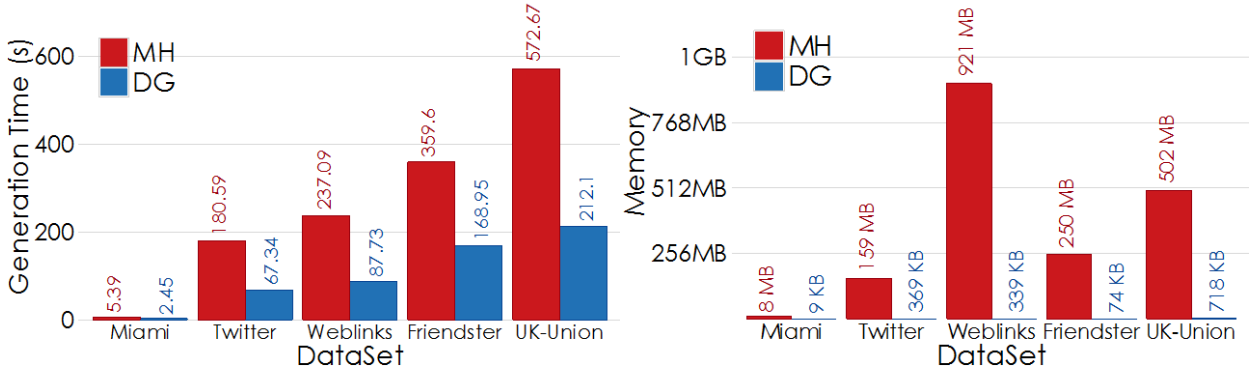


Figure 4-4 Runtime and memory requirement comparison between our DG algorithm and the previous MH algorithm [45] on several datasets.

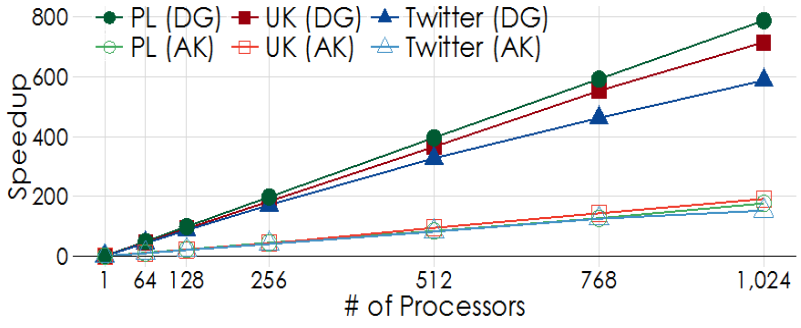


Figure 4-5. Strong scaling of our parallel DG algorithm and the previous AK algorithm [4] on several datasets.

Figure 4-5 shows the speedup of our parallel DG algorithm along with the best-known previous parallel algorithm [4] (referred as the AK algorithm) for a massive power-law (PL) and two large real-world graphs (Twitter and UK-Union). Speedups are measured as T_s / T_p, where T_s and T_p are the running time of the

sequential and the parallel algorithm, respectively. The number of processors is varied from 1 to 1024. As shown in Fig. 4-7, our algorithm achieves almost linear speedup for each graph. The AK algorithm also has a linear speedup. However our algorithm is approximately four times faster than the AK algorithm. Moreover, our algorithm requires less memory (O(∧) memory) than the AK algorithm (O(*n*) memory). For example, for the Twitter, UK-Union, and PL graphs, the DG algorithm takes about 440, 716, and 16000 times less memory than the AK algorithm, respectively.

4.5. SPIDAL Algorithms – Triangle Counting

Motivation: Counting triangles in a graph is a fundamental and important algorithmic problem in graph analysis, and its solution can be used in solving many other problems such as the computation of clustering coefficient, transitivity, and triangular connectivity [39, 46]. Existence of triangles and the resulting high clustering coefficient in a social network reflect some common theories of social science, e.g., homophily where people become friends with those similar to themselves and triadic closure where people who have common friends tend to be friends themselves [47]. Further, triangle counting has important applications in graph mining such as detecting spamming activity and assessing content quality in social networks [48] and detecting communities or clusters in social and information networks [49].

Finding the number of triangles in a network (graph) is an important problem in mining and analysis of complex networks. Triangle counting and enumeration is an important special case of subgraph mining. Specialized algorithms for this problem can outperform the general algorithms for the subgraph analysis problem significantly.

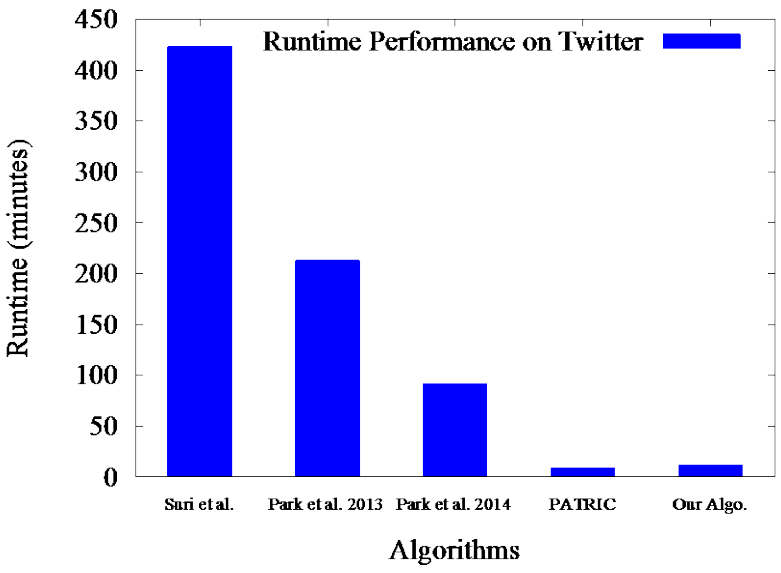


Figure 4-6. Runtime comparison of various algorithms for counting triangles

There are few Hadoop-based distributed-memory parallel algorithms (by Suri et al. and Park et al.) for counting triangles. These algorithms generate huge volumes of intermediate data for shuffling and regrouping, which require a large amount of time and memory. Previously we have developed an MPI-based distributed-memory parallel algorithm, called PATRIC

[50], which uses overlapping partitioning of the given graph. Although PATRIC significantly improves both time and space requirement comparing to the Hadoop-based algorithms, it still requires large memory per processor due to overlapping partitions.

Very recently we have developed a space-efficient MPI-based parallel algorithm for counting and enumerating triangles in massive networks. The algorithm divides the network into non-overlapping partitions. Experimental results shown in figure 4-6, on some real-world networks demonstrate up to 25-fold space saving over the previous algorithm PATRIC, while the runtime is comparable to that of PATRIC. For example, for the Twitter network, our space-efficient require 265MB memory per processor in contrast to 4254MB per processor for PATRIC. Figure 4 shows runtime comparisons on Twitter network for these algorithms.

4.6. SPIDAL Algorithm – Community Detection

A community in a network is a group of nodes such that the nodes within the community are densely connected but there are fewer edges from these nodes to the nodes outside the community. Complex networks generally consist of communities or clusters of nodes, each having distinct role or function. Each functional unit (community) appears as a tightly-knit set of nodes having higher connection inside the set than outside. Finding communities may reveal the organization of complex systems and their function. For instance, communities are often interpreted as organizational units in social networks, functional units in biological networks, or scientific disciplines in citation networks. Thus detecting Communities in massive networks such as emerging social and information networks has become an interesting and fundamental problem in network science.

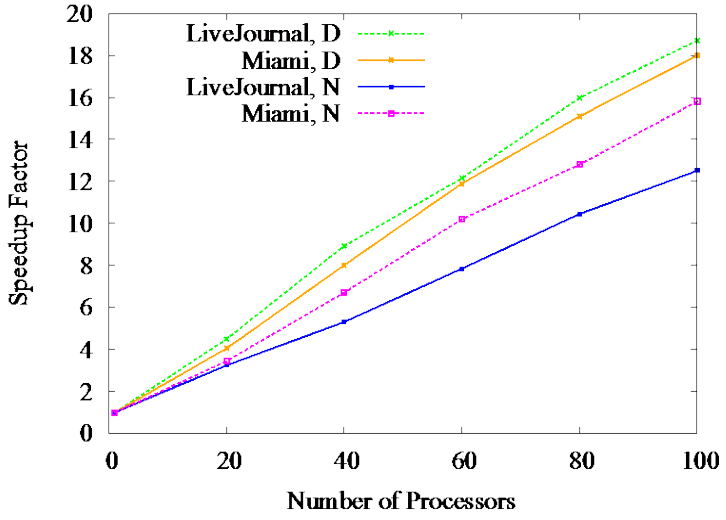


Figure 4-7. Strong scaling of our parallel community detection algorithm on two networks: LiveJournal and Miami social contact network. The results are shown for two partitioning schemes: based on the number of nodes (N), based on the degrees of the nodes (D).

Although a fairly large volume of work addressed the sequential algorithms for community detection (see a survey in [51]), only recently has attention been given to parallel algorithms. There are some existing parallel algorithm for shared-memory [52], Bulk Synchronous Parallel (BSP) [53], GPU [54], and MapReduce [55] frameworks. We propose an MPI-based distributed-memory parallel algorithm that later will be implemented with Harp. By partitioning the graph, this algorithm allows us to work with large-scale graphs and scales well with the increasing number of processors. This is an ongoing work. Below we present some of our preliminary experimental results. Figure 4-7 shows the speedup factor of our algorithm with the increasing number of processors on two networks.

4.7. SPIDAL Algorithms – Core

The initial contributions to SPIDAL are available from Github [56] and are described here and are already given in Table 4-4. As MIDAS and SPIDAL development proceeded in parallel, much of initial SPIDAL work used different technology from that highlighted in MIDAS; for example, not all routines are available in Harp or use the SPIDAL Java optimizations. As this project matures and optimized MIDAS components become available, we will re-engineer current SPIDAL accordingly. Further based on input from users of MLlib and Mahout, we will plan a design process to ensure a uniform consistent programmatic interface to the SPIDAL library.

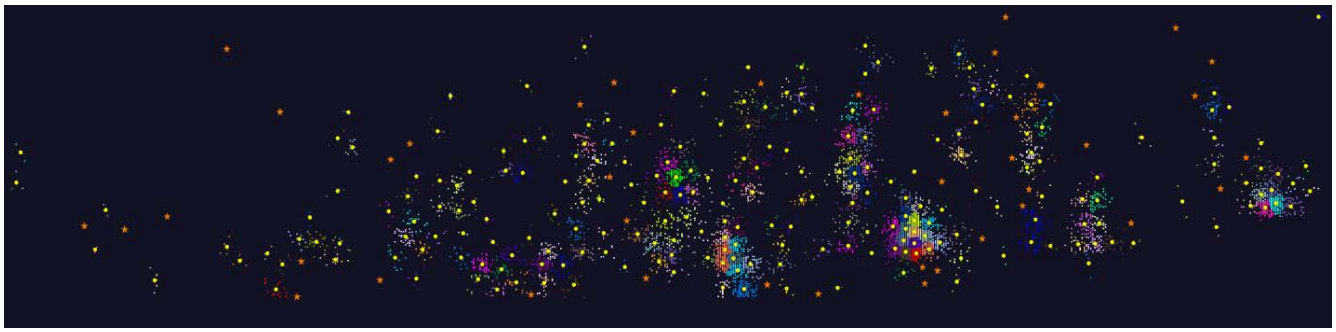


Figure 4-8: Visualization using WebPlotViz of a small part (<0.1%) of a clustering using DA-VS of 10.9 million LCMS peaks into 423400 clusters extending work in [57]. Orange stars are the 1% of points outside clusters, yellow circles cluster centers, colored dots are clustered peaks.

We now describe status of several core machine learning routines.

1. **O(N²) distance matrices** calculation with Hadoop parallelism and various options such as storage in MongoDB or distributed files, normalization, packing to save memory usage, and calculation exploiting symmetry. This is built into many existing approaches and described in section 5.5. We will separate this and make it a separate SPIDAL library.

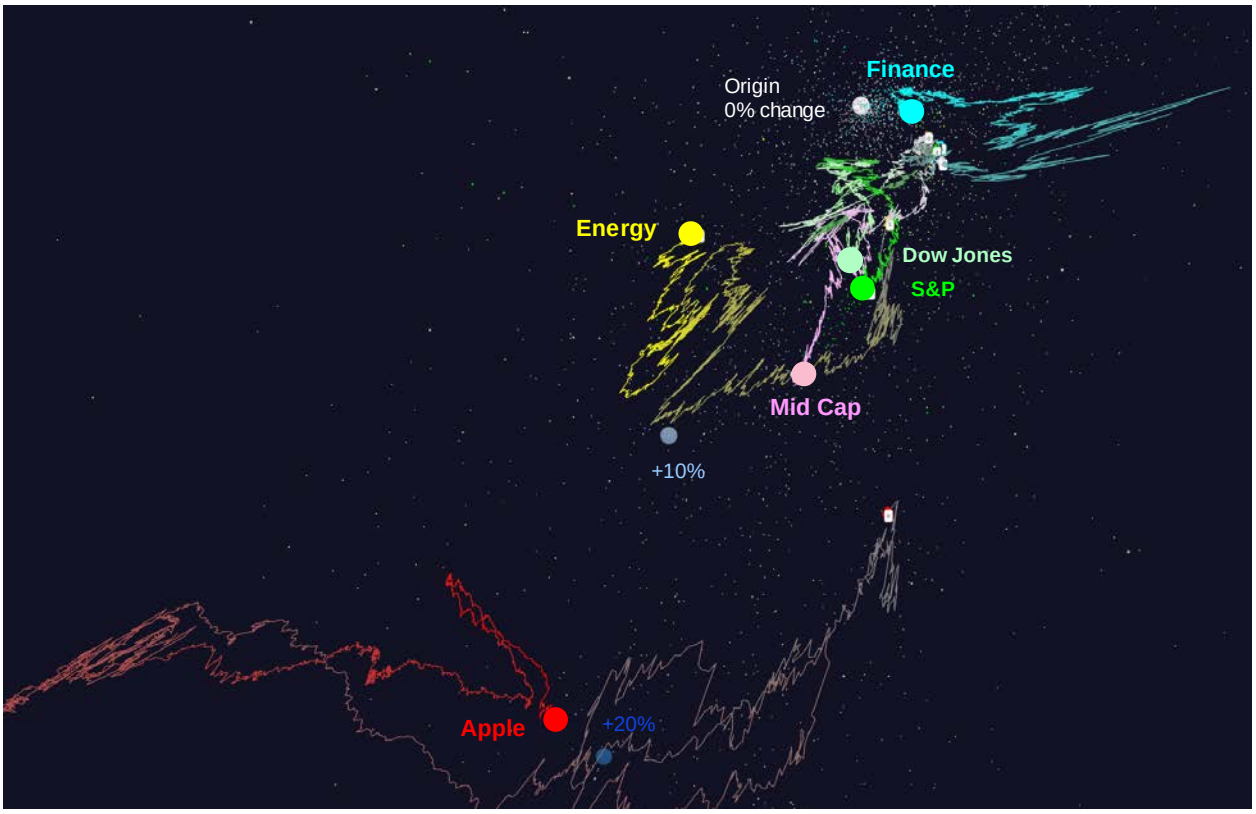


Figure 4-9: Trajectories of values of 6 financial instruments (stocks with ETF's) using one day values measured from January 2004 and ending December 2015. Filled circles are final values and 6000 stocks are used in the DA-MDS 3D projection of vectors of daily stock values. [58]

2. **WDA-SMACOF or DA-MDS: Multidimensional scaling MDS** is optimal nonlinear dimension reduction enhanced by SMACOF, deterministic annealing and Conjugate gradient for non-uniform weights [59]. Used in many applications [2] and is illustrated in figures 4-9 to 4-12. This is believed to be ey most accurate non-linear dimension reduction routine and the only one whose performance scales to large parallel machines. It supports Sammon's method and missing distance measurements. There are MPI and MIDAS (Harp) versions.

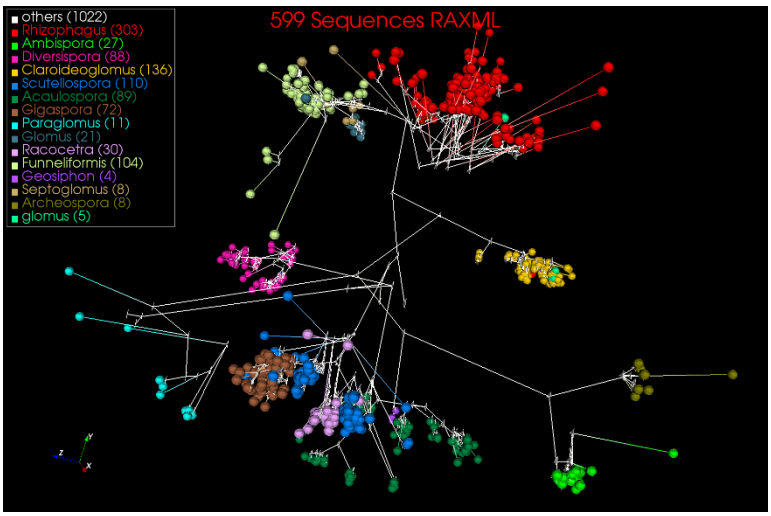


Figure 4-10: 3D Phylogenetic Tree from [2, 3] using DA-MDS and WebPlotViz

3. **MDS Alignment** to optimally align related point sets, as in MDS time series

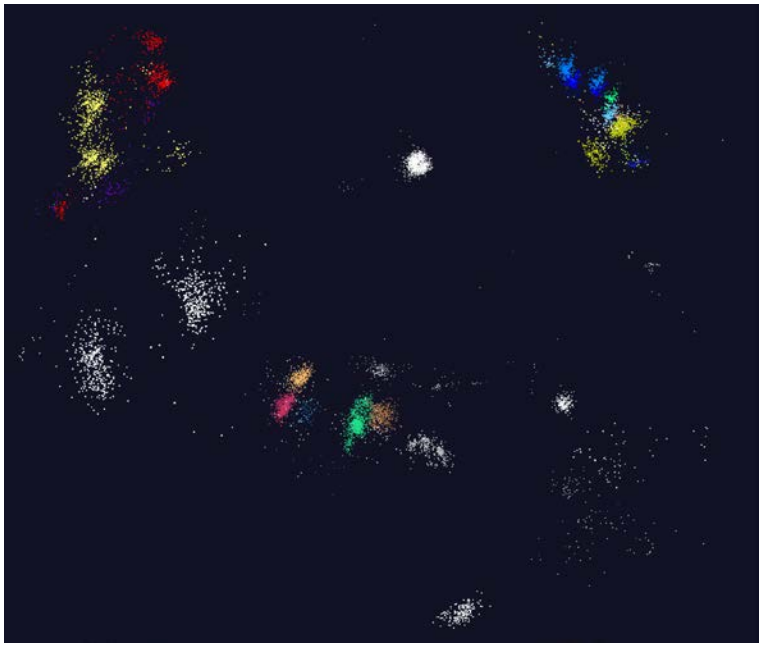


Figure 4-11: Results from [1] using DA-MDS to visualize in WebPlotViz for a set of Fungi sequences here colored by identified species

4. **WebPlotViz** data management (MongoDB) and browser visualization for 3D point sets including time series. Available as source or SaaS

5. **MDS as χ^2 using Manxcat** (see next section on optimization). Alternative more general but less reliable solution of MDS. [60, 61]Latest version of WDA-SMACOF usually preferable and our use of this has declined.

6. **Other Dimension Reduction**: SVD, PCA, GTM algorithms are understood but no work has been done within SPIDAL.

7. **DA-PWC Deterministic Annealing Pairwise Clustering** for case where points aren't in a vector space; used extensively to cluster DNA and proteomic sequences; improved algorithm over other published. Parallelism good but needs SPIDAL Java.

8. **DAVS Deterministic Annealing Clustering for vectors**; includes specification of errors and limit on cluster sizes. Gives very accurate answers for cases where distinct clustering exists. Being upgraded for new LC-MS proteomics data with one million clusters in 27 million size data set shown in figure 4-8.

9. **K-means basic vector clustering**: fast and adequate where clusters aren't needed accurately

10. **Elkan's improved K-means vector clustering**: for high dimensional spaces; uses triangle inequality to avoid expensive distance calculations

The above 10 routines are complete usable parallel algorithms and have good parallel performance although they still need extensive work on use of MIDAS, SPIDAL Java and establishing good uniform interface. Parallel implementations are needed for logistic regression, Random Forest, SVM, Collaborative Filtering, TF-IDF search and other Spark MLlib and Mahout algorithms. These are typically simpler than codes already implemented but represent a major software engineering and performance tuning project.

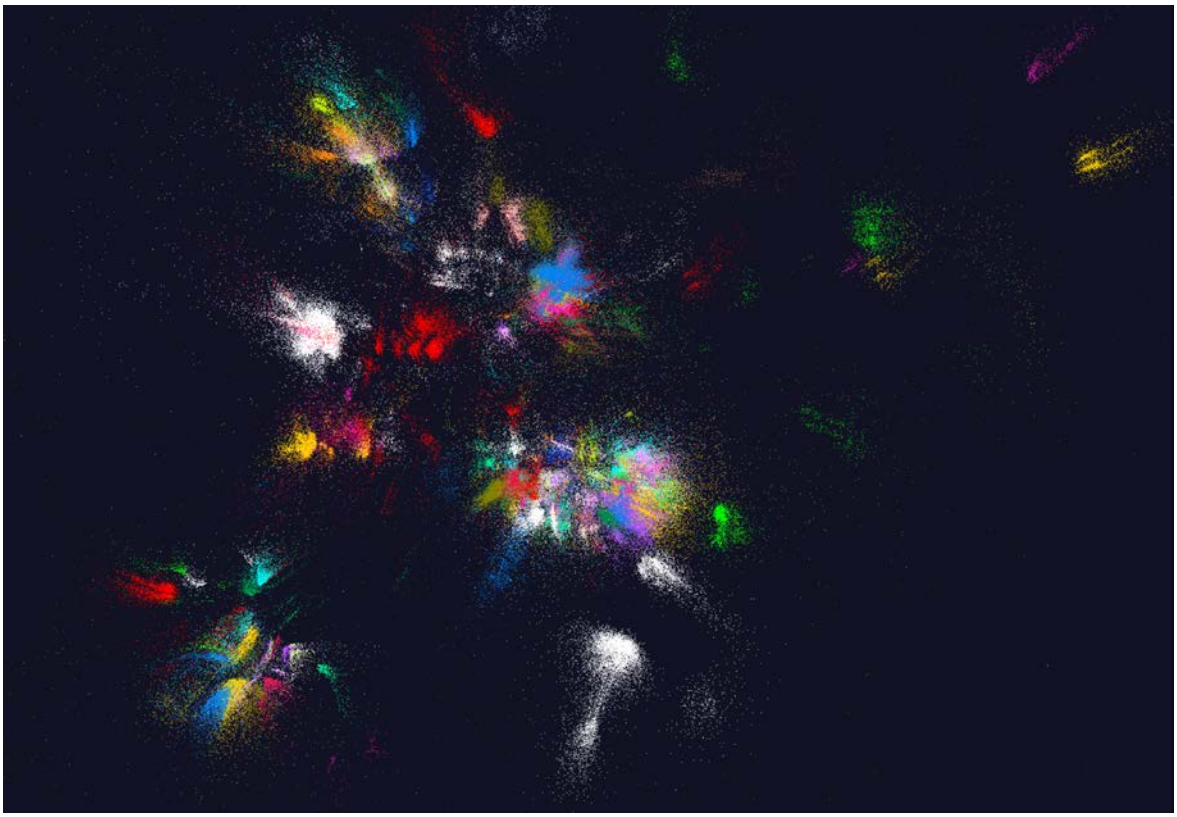


Figure 4-12: Early studies for [1, 2] using DA-MDS to visualize in WebPlotViz for a set of Fungi sequences with 127 clusters determined by DA-PWC

4.8. SPIDAL Algorithms – Optimization

Many problems in computer science and science in general can be posed as mathematical optimization tasks, where the goal is to find the values for a set of variables that minimize a given *objective* or *energy* function of those variables. In applications, these tasks often arise in the context of fitting a model to data. For instance, nearly all machine learning algorithms are simply optimizing a set of model parameters by minimizing an objective function that measures the error of the model on a set of labeled training examples. Meanwhile data mining algorithms like K-means and LDA are similarly fitting model parameters to data to minimize some residual error. Many of the imaging applications in Section 5, for example, are simply finding a "simple" model to explain

complicated image data, e.g. a model that compactly describes a noisy radar echogram in terms of ice layer structure, or a segmentation that compactly describes a high-resolution CT scan.

Optimization problems can often be classified into various broad categories depending on the form of the objective function, the domain of the unknown variables, and the type of solution that is required. While there is no efficient algorithm that can solve all optimization problems, algorithms do exist for many of these general categories, and we are integrating several of the most common in SPIDAL.

Continuous optimization problems. For the common class of optimization problems where the variables are continuous and either the objective function is convex or a local minimum (as opposed to a global minimum) is acceptable, we are implementing **Manxcat,** a Levenberg Marquardt Algorithm for non-linear χ^2 optimization. This algorithm uses a sophisticated version of Newton's method calculating value and derivatives of the objective function. It is parallelizable in both the calculation of the objective function and in the parameters to be determined. We have completed the implementation of this algorithm but it still needs to be optimized for SPIDAL in Java.

Discrete optimization problems commonly arise in computer vision, language modeling, operations research, and other applications. Since in general discrete optimization is NP-hard, various efficient algorithms have been developed for objective functions with specific forms. Other algorithms can produce approximate but typically high-quality solutions even for some NP-hard problems. We have implemented several of these algorithms and are working to integrate them into SPIDAL. In particular:

- **The Viterbi** algorithm finds the maximum a posteriori (MAP) solution for a Hidden Markov Model (HMM), which has an objective function that can be written as a sum of pairs of variables, such that the graph of these pairs is acyclic. The running time is O(n*s^2) where n is the number of variables and s is the number of possible states each variable can take. We will provide an "embarrassingly parallel" version that processes multiple problems (e.g. many images or many sentences) independently. Because Viterbi is so efficient, we do not believe parallelizing within the same problem is needed in our application space.
- **Forward-backward algorithm** computes marginal distributions over HMM variables, with similar characteristics as Viterbi.
- **Loopy belief propagation (LBP)** for approximately finding the maximum a posteriori (MAP) solution for a Markov Random Field (MRF). An MRF is a generalization of an HMM in which the objective function is still a sum over functions of pairs of variables, but this

pairwise relationship structure does not necessarily form a tree. Here the running time is O(n^2*s^2*i) in the worst case where n is number of variables, s is number of states per variable, and i is number of iterations required (which is usually a function of n, e.g. log(n) or sqrt(n)). Here there are various parallelization strategies depending on values of s and n for any given problem. We will provide two parallel versions: embarrassingly parallel version for when s and n are relatively modest, and parallelizing each iteration of the same problem for the common situation when s and n are quite large so that each iteration takes a long time relative to the number of iterations required.

- **Markov Chain Monte Carlo (MCMC)** for approximately computing marginal distributions and sampling over MRF variables. Similar to LBP with the same two parallelization strategies.

The first – Manxcat – is complete as a library member but needs SPIDAL Java optimization and other packaging such as improved interface. The other four exist in application code but need to be abstracted as general library members with software engineering and performance work.

4.9. SPIDAL Algorithms – Polar Remote Sensing Algorithms

Motivation: This extends earlier successful work on 2D image processing to a 3D formulation that demonstrates how linking multiple images together will produce more reliable results from constraint of smoothness between images. This motivates some of our SPIDAL optimization algorithms.

We are investigating radar informatics and image processing to reconstruct 3D ice structure in polar regions using novel analysis and processing of data from CReSIS radar systems. As it flies along a flight path, the CReSIS radar data processing outputs a sequence of tomographic cross sections or "slices" that characterize the returned radar signals at different orientations with respect to the vehicle (Figure 4-13) [62]. These slices are 2D images, in polar coordinates, where each pixel represents the energy return at the corresponding angle (with respect to the radar device) and radius (termed range in radar nomenclature). However, these ranges and angles are inferred by complex radar array processing, and can be noisy and imprecise when the desired target signal is weak relative to the noise from electronics and interfering scatterers. We have been developing optimization algorithms that integrate this noisy evidence with known constraints, like smoothness of the ice structure, in order to produce more accurate tomographic slices and in turn more accurate 3D reconstructions.

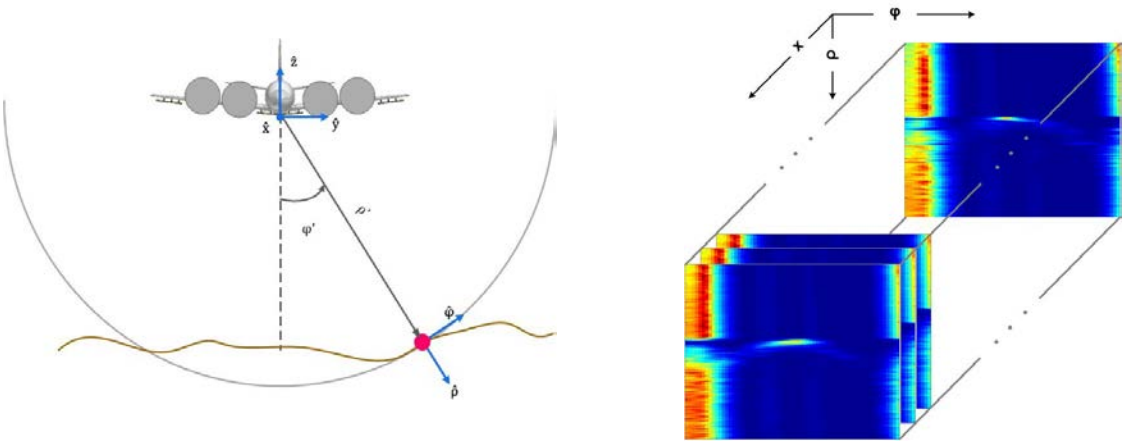


Figure 4-13: At each position along the flight track, the radar gives a cross-section view, parameterized by angle and range, of the ice structure, which yields a set of 2-d tomographic slices (right) along the flight path.

In particular, we pose the problem of estimating true tomographic slices from noisy ones as a Markov Random Field model. We applied similar MRF models in our earlier work on finding ice layer boundaries in 2D radar echograms [63, 64]. We use Loopy Belief Propagation (LBP) and Markov Chain Monte Carlo (MCMC) [65] to perform inference on these models. Although both MCMC and LBP are approximate inference algorithms (since exact inference is NP-hard), they have proven to be successful in a wide range of vision problems, and we find that they also work well for ours.

We have been testing the algorithm using a simulator created by CReSIS that lets us take a ground-truth 3D bed structure, create synthetic tomographic images corresponding to that structure with user-controllable noise parameters, and then run our algorithm to compare the output to ground truth. We also compare to the initial simpler algorithm that CReSIS had developed [62], which was based on a simple interpolation of the array processor. The array processor uses a local optimization (single pixel) based on the maximum likelihood estimator. Figure 4-14 shows sample results.

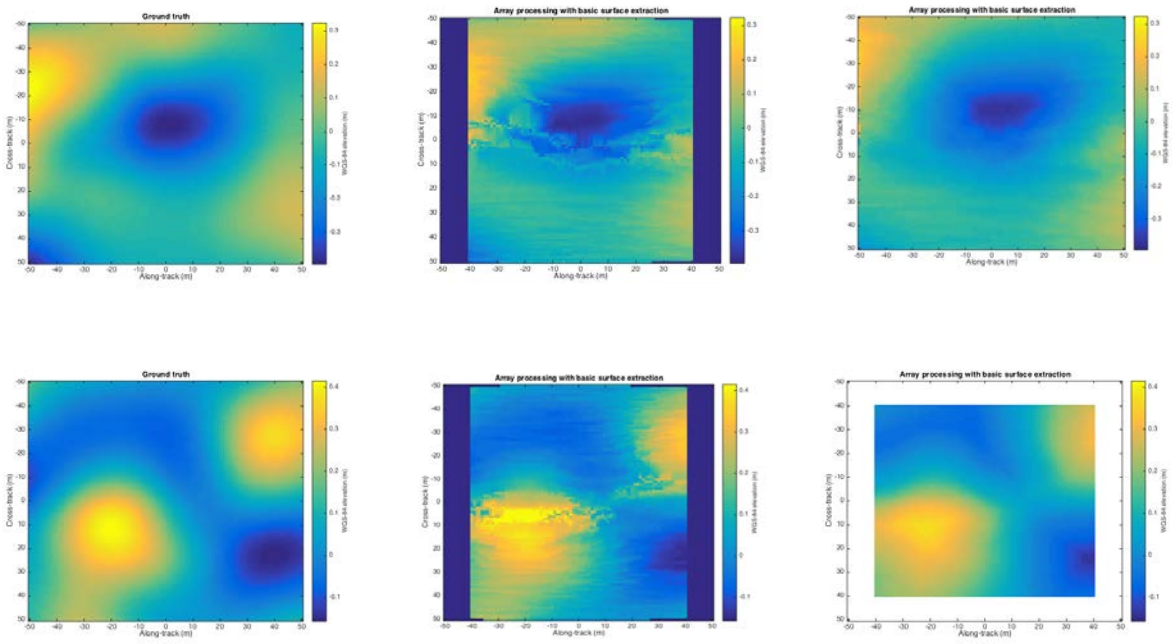


Figure 4-14: Two sample results (one per row) for reconstructing bedrock in 3D, each showing (left) ground truth, (center) prior algorithm based on a simple interpolation of maximum likelihood estimators, and (right) our technique based on a Markov Random Field formulation. Each image represents a 3D depth map, with along track and cross track dimensions on the x-axis and y-axis respectively, and depth coded as colors. Note that the new algorithm at right more cleanly and faithfully reconstructs the ground truth at left.

As part of this effort, we are also investigating novel probabilistic techniques for understanding the along-path cross sections of the radar signals. Figure 4-15 (left) shows an example of a cross section generated by the radar processing, where the yellow line in the middle corresponds to the bedrock signal that we would like to infer, but is weak and incomplete. As in this example, these signals are noisy, with many other confusing lines with similar structure. Fortunately, external information (like known prior properties of ice sheets and evidence from previous and subsequent cross sections) can be used to resolve these difficulties. We pose the problem again in terms of inference on a probabilistic graphical model and again apply techniques based on belief propagation and MCMC. Figure 4-14 shows sample results.

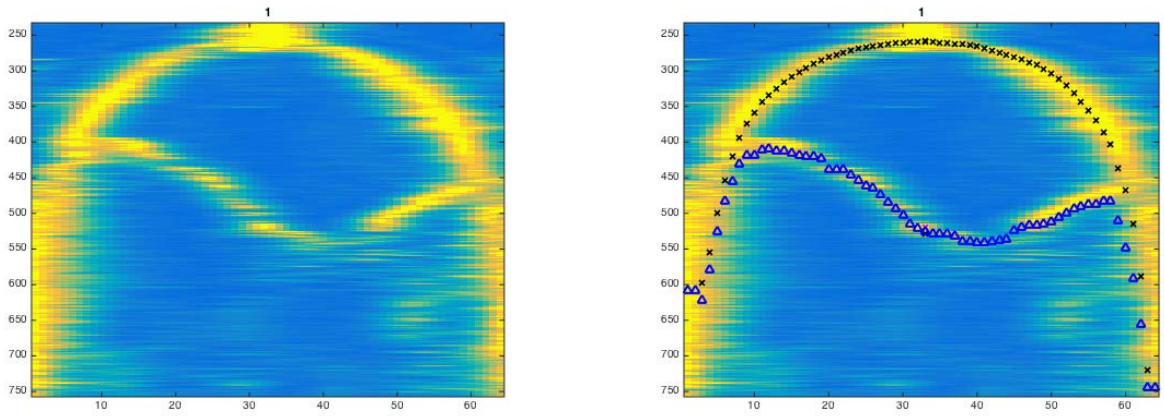


Figure 4-15: Sample visualization of cross section from CRESIS radar, (left) before our analysis and (right) after localization of the bedrock layer. The x-axis corresponds to the cross track dimension and the y-axis corresponds to reflectance time (which approximately correlates with depth).

4.10. SPIDAL Algorithms – Nuclei Segmentation for Pathology Images

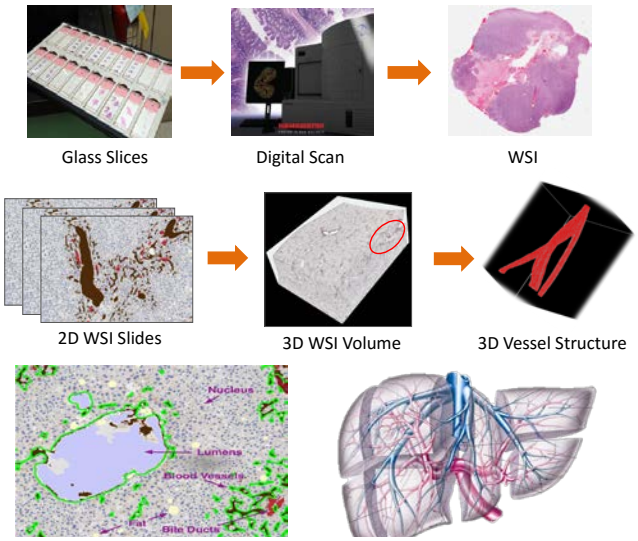


Figure 4-16: 2D and 3D pathology image analysis

Motivation: High-throughput digital scanning technologies, illustrated in Figure 4-16, have turned pathology image data into an emerging imaging modality to facilitate basic scientific research and diagnosis at cellular level by clinicians and biomedical researchers. With quantitative analysis of pathology imaging data, clinicians and researchers are able to explore the morphological and functional characteristics of biological systems as well as gain insight on the underlying mechanisms of normal tissue development and pathological evolutions of distinct diseases. Quantitative analyses of pathology images include segmentation of micro-anatomic objects such as nuclei and extraction of image features such as area, perimeter, and eccentricity [66]. Recently, 3D digital pathology was made possible through slicing tissues into serial sections. By registering consecutive slices, segmenting and reconstructing 3D micro-anatomic objects, it is possible to provide a 3D tissue view to explore spatial relationships and patterns among micro-anatomic objects to support biomedical research [67, 68]. For example, liver disease diagnosis and analytics rely on 3D structural features of blood

vessels and their 3D spatial relationships with cells [69]. The information-lossless 3D tissue view with microscopy imaging volumes holds significant potential to enhance the study of both normal and diseased processes, and represents a new frontend for digital pathology [70].

There are major research challenges to detect and quantify spatial clusters at extreme scale due to the explosion of spatial data at micro-anatomic level and patient level. First, we will need to develop or adapt spatial clustering methods that can support detection of spatial clusters of complex shapes such as pseudopalisades. Second, we will need to make existing spatial clustering methods highly scalable for spatial big data. Stony Brook University will collaborate with Indiana University on developing scalable spatial clustering methods to support biomedical informatics problems driven by pathology imaging and GIS oriented public health studies.

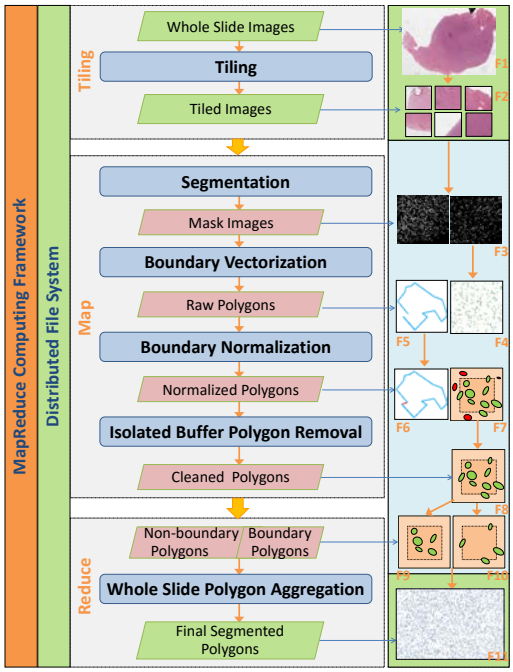


Figure 4-17: MapReduce based pipeline for nuclei segmentation

2D pathology image analysis: We have developed a novel and robust segmentation method for 2-D cells in histopathologic images [71]. It consists of a new seed detection algorithm and a newly designed cell contour deformation method based on a sparse shape prior guided variational level set framework. The cell seed detection algorithm draws joint information of spatial connectivity, distance constraint, image edge map, and a shape-based voting result derived from eigenvalue analysis of Hessian matrix across multiple scales. Thus, it produces robust and accurate seed detection results, especially for overlapped or occluded cells. With cell contours initialized from these seeds, We deform them within a variational level set framework where we aim to minimize a new energy functional that incorporates a shape term in a sparse shape prior representation, an adaptive contour occlusion penalty term, and a boundary term encouraging contours to converge to strong edges. As a result, this approach is able to accommodate mutual occlusions and detect contours of multiple intersected objects simultaneously. This work will be useful for investigations on the influence of hypoxia and transcription factor expression on the orientation of tumor cells to assess their direction of migration. To support scalable pathology image analysis, we develop a MapReduce based framework to parallelize the image analysis pipelines, which includes multiple steps: tiling of whole slide images, segmentation of objects for each title, boundary normalization

for boundary-crossing objects, and aggregation of final spatial objects such as nuclei. A workflow is shown in Figure 4-17.

3D pathology image analysis: Additionally, we have made substantial progress on quantitative reconstruction of 3D blood vessel structures with microscopy images of serial tissue sections [68, 70]. This work will be particularly useful for investigations of the spatial configurations and signaling networks of tumor cells for creating a pro-angiogenic environment. Specifically, we have developed a fully automated framework for 3D vessel reconstruction with a set of histological whole-slide images of sequential tissue sections. We have managed to segment cells, vessels, and lumens with a morphology reconstruction segmentation method applied to image channels associated with different stains (i.e. Haematoxylin and DAB) decomposed with the color deconvolution technique. All slides in the same series are registered with rigid registration at low resolution and non-rigid registration on small image patches with a cubic B-Spline transform. We associated the segmented vessel objects across all slides by local bi-slide vessel mapping and global vessel structure association. Bi-slide vessel mapping generates sub-vessel structures between adjacent slides with pre-defined one-to-one, one-to-two, one-to-none and none-to-one association cases. In the global association, a Bayesian Maximum a Posteriori (MAP) framework was adopted to recover the global vessel structures across all images with the posterior probability modeled as a Markov chain. To group vessel cross-sections from different slices, we have used the motion and shape information with Kanade-Lucas-Tomasi Feature Tracker, morphology features, and distribution of Hausdorf distances. For better visualization, we uniformly sample vessel boundary points based on arc length from each 2D slice and interpolate vessel boundaries between the given slices. The isosurface of the analyzed vessel data volume is computed and rendered to show the 3-D structure. The proposed 3D vessel analysis framework is generic and can be readily applied to the analytics of other 3D biological entities of common interest in other biomedical investigations.

4.11. SPIDAL Algorithms – Spatial Querying Methods

Motivation: Over recent years the proliferation of mobile phones, Internet of things (IoT) projects and ubiquitous sensory measurement technologies have contributed to generating multidimensional spatial data at an unprecedented scale and rate. Furthermore, collaborative spatial data collection projects, such as OpenStreetMap, have sped up the process in many folds. Spatial data collection, storage, querying and analysis have become increasingly important for scientific and business as well as daily user applications. Analysis of this plethora of spatial data involves complex queries such as spatial joins or spatial cross-matching, overlay of multiple sets of spatial objects, spatial proximity computations between objects, and queries for global spatial pattern discovery. In addition, support of high performance spatial queries on large volumes of 3D spatial data is becoming increasingly relevant in various emerging scientific applications, which are increasingly data- and compute-intensive. In particular, 3D analytical pathology imaging provides high potential to support image based computer aided diagnosis, and quantitative analysis of large-scale 3D pathology image volumes generates tremendous amounts of spatially derived 3D micro-anatomic objects, such as 3D blood vessels and nuclei. Spatial exploration shown in Figure 4-18, of such massive 3D spatial data requires effective and efficient querying methods. However, there are major challenges to support spatial queries: the "big data" challenge due to explosion of spatial data, the complex spatial object representation, and the high geometric computation complexity.

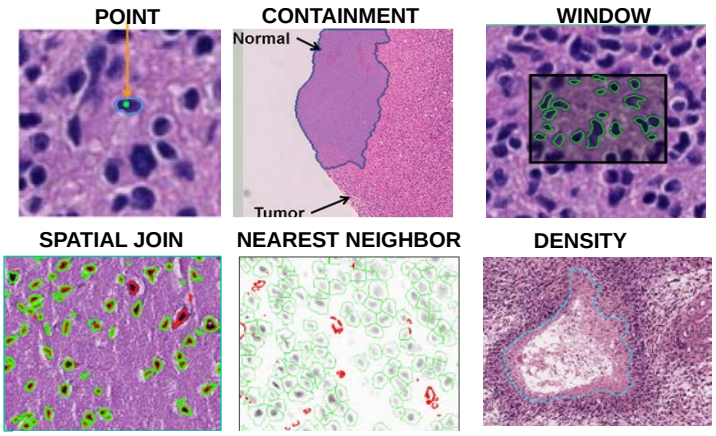


Figure 4-18: Example spatial queries for digital pathology imaging

SparkGIS: We have developed a Spark based spatial querying method, which ports spatial querying libraries in Hadoop-GIS to run on Apache Spark framework [72]. SparkGIS removes HDFS dependency and supports multiple types of data storage, such as MongoDB, HDFS, and local file systems. By taking advantage of in-memory computing, SparkGIS significantly reduces I/O cost and boosts query performance. SparkGIS also supports streamed data processing, and can process data without waiting for all the data to be ready. SparkGIS supports common spatial queries, including spatial joins, containment queries, and nearest neighbor queries. SparkGIS also

supports plugins into the querying pipelines, for example, computing statistics on top of querying results, and having them integrated into a single job. SparkGIS has been tested and deployed to support evaluation of segmentation results for large scale pathology image analysis.

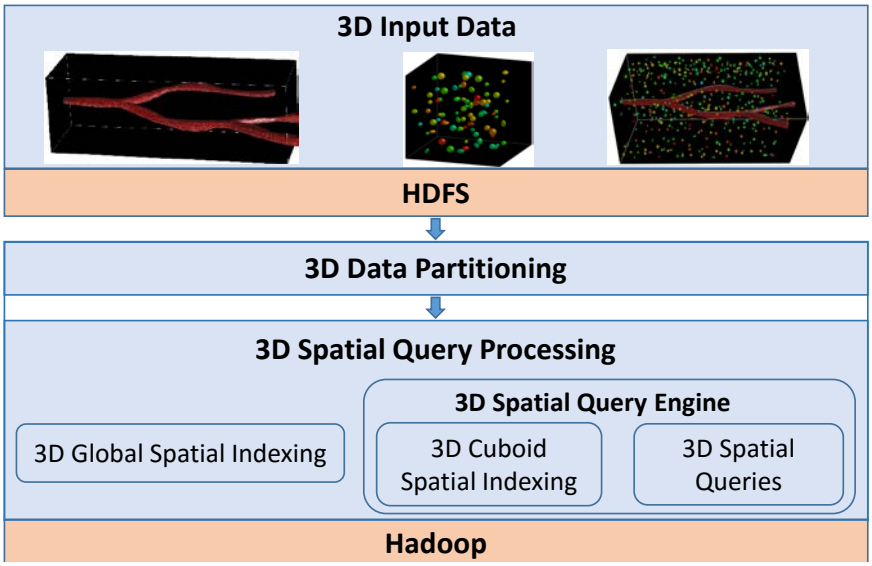


Figure 4-19: Architecture Overview of Hadoop-GIS 3D

Hadoop-GIS 3D: We have developed a scalable and efficient 3D spatial query system for querying massive 3D spatial data based on MapReduce [70] shown in Figure 4-19. Our system supports multiple types of spatial queries on MapReduce through 3D spatial data partitioning, customizable 3D spatial query engine, and implicit parallel spatial query execution. The system utilizes multi-level spatial indexing to achieve efficient query processing, including global partition indexing for data retrieval and on-demand local spatial indexing for spatial query processing. Our prototype system supports two typical queries: 3D spatial joins and 3D K-nearest neighbor queries. Our experiments demonstrate the high efficiency and scalability of the system to support 3D spatial queries on 3D micro-anatomic objects for analytical pathology imaging on MapReduce.

5 Applications

5.1. Summary

5.2. Overview of Imaging Applications

5.3. Enabled Applications – Digital Pathology

5.4. Enabled Applications – Public Health

5.5. Enabled Applications – Biomolecular Simulation Data Analysis

5.1. Summary

As explained in the introduction, our project uses applications to motivate and test the proposed building blocks. These applications are described in this section and earlier in Section 4 where application and algorithm were intertwined. There are also applications not funded by the project that are helpful for the building blocks.

The project applications involve multiple examples of image-based data, a general point described in Section 5.2. These are seen in remote sensing (KU) application of Section 4.9 and the digital pathology (SB) case in Section 5.3. There are interesting synergies between geospatial information GIS problems and the large 2D and 3D images seen in pathology and this is explained in public health (SB) application in Section 5.4. The last subsection 5.5 in this section describes the analysis of biomolecular simulations (Utah, ASU, Rutgers). Section 4 describes graph algorithms identified by Virginia Tech from their study of networks and the CINET infrastructure [73, 74] which is a resource we will use to disseminate SPIDAL. IU and Rutgers have substantial work on streaming applications discussed in Section 7.3. Recently early HPC-ABDS work at IU helped the online Twitter streaming data repository Osome go live [75, 76]. Other IU applications driving SPIDAL and MIDAS are bioinformatics, financial informatics and robotics.

5.2. Overview of Imaging Applications

We motivate and test our building blocks through multiple applications of image processing and computer vision. Although these applications are from very different areas, they share many of the same core algorithms, allowing them to readily apply the same set of basic building blocks described above. In particular:

Radar informatics. Our collaboration with CReSIS is developing algorithms for automated and semi-automated analysis of large-scale data produced by radar sensing of the Earth's polar regions, as we described in Section 4.8. Although these radar datasets are not images *per se,* they are 2D and 3D data that can be readily visualized and represented as images (e.g. radar echograms), which let us apply the same basic building blocks as with traditional visual-spectrum image processing. For example, finding layers of ice in these echograms is an image segmentation problem [77], which in turn can be formulated in terms of an energy minimization problem that tries to fit simple models to the data [63, 64]. This energy minimization problem can be solved using the various optimization algorithms described above, including the Viterbi algorithm, Loopy Belief Propagation, and gradient descent.

Online social images. Computer vision is a very active research area, with most current research focused on extracting semantic information from consumer-style images, like those uploaded to photo sharing sites like Flickr and Facebook. Nearly all work in this area uses machine learning to automatically train classifiers for various tasks of interest to consumers, like face detection [78], image captioning [79], scene recognition [80], etc. For instance, Support Vector Machines [81] and Convolutional Neural Networks [82] are among the most popular learning techniques, and both are formulated in terms of energy minimization problems for which our optimization algorithms may potentially be applied. After models are trained, the main computation challenge with these datasets is their enormous quantity, typically millions to billions of images. However, the images can be processed independently, thus taking advantage of the pleasingly parallel versions of our building block implementations.

Pathology and remote sensing. In contrast to social images, pathology and remote sensing images may be relatively few in quantity but enormous in size. For example, each pathology image could have 10 billion pixels, and we may extract a million spatial objects and 100 million features (dozens to 100 features per object) per image. We often tile the image into 4K x 4K tiles for processing. While the tasks involved in processing these images are similar to social images (e.g. segmentation, recognition, etc.), the huge image size requires developing buffering-based tiling to

handle boundary-crossing objects. For each typical research study, we may have hundreds to thousands of pathology images.

5.3. Enabled Applications – Digital Pathology

Overview: Digital pathology images scanned from human tissue specimens provide rich information about morphological and functional characteristics of biological systems. Pathology image analysis has high potential to provide diagnostic assistance, identify therapeutic targets, and predict patient outcomes and therapeutic responses. It relies on both pathology image analysis algorithms to extract spatial information from images and spatial querying methods to explore spatial relationships or spatial patterns for micro-anatomic objects. Digital pathology includes both 2D pathology imaging and 3D pathology imaging.

2D Digital pathology: 2D digital pathology images are generated through scanning human tissue specimens with high resolution microscope scanners. Examination of high resolution whole slide images enables more effective diagnosis, prognosis and prediction of cancer and other complex diseases. Analytical pathology imaging provides quantitative methods to derive tremendous amounts of spatial data about micro-anatomic objects [66]. Indeed, 2D pathology image analysis has been extensively used to support biomedical research for various diseases [66, 83], which produces 2D geometric objects representing cells, nuclei, and blood vessels, among others.

3D Digital Pathology: 3D digital pathology works through slicing tissues into serial sections. By registering consecutive slices, segmenting and reconstructing 3D micro-anatomic objects, it is possible to provide a 3D tissue view to explore spatial relationships and patterns among micro-anatomic objects to support biomedical research [67, 68]. For example, liver disease diagnosis and analytics rely on 3D structural features of blood vessels and their 3D spatial relationships with cells [69]. The information-lossless 3D tissue view with microscopy imaging volumes holds significant potential in terms of studying processes for both healthy and ill samples, and represents a new frontend for digital pathology [70].

Quantitative analysis of 2D digital pathology images relies on image segmentation and feature extraction for 2D images. 2D analysis involves image registration, 2D segmentation, 3D object association, 3D object interpolation, and 3D object representation and visualization. Once spatial data is derived, spatial methods will be applied for spatial data exploration. Image analysis methods such as nuclei segmentation and spatial querying methods such as spatial joins could be parallelized through MapReduce or Spark.

5.4. Enabled Applications – Public Health

Overview: GIS-oriented public health research has a strong focus on the locations of patients and the agents of disease, and studies the spatial patterns and variations.

Integrating multiple spatial big data sources at fine spatial resolutions allow public health researchers and health officials to adequately identify, analyze, and monitor health problems at the community level. This will rely on high performance spatial querying methods on data integration of multiple spatial data sources.

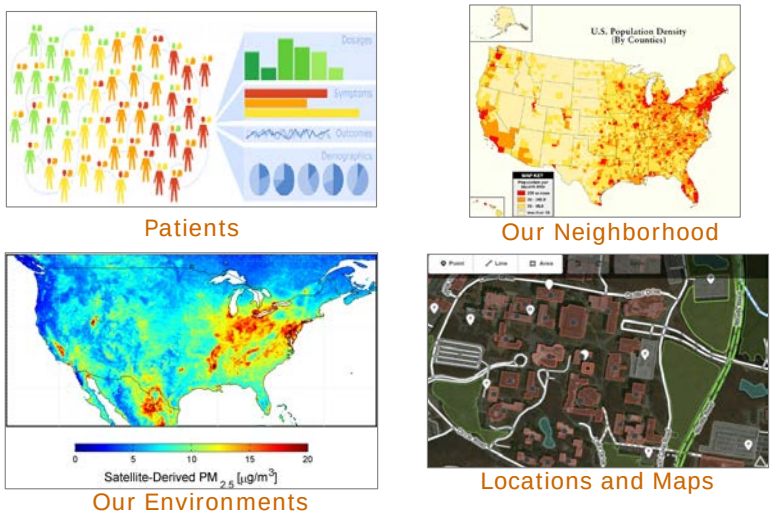


Figure 5-1: Spatial Big Data Analysis for Public Health

Integrative Spatial Big Data Analytics for Public Health: GIS oriented public health research illustrated in Figure 5-1, has a strong focus on the locations of patients and the agents of disease, and studies the community and region level patterns and variations, as well as the impact of demographical, socio-economical, and environmental factors on diseases and human health. In the past, due to limited accessibility of health outcome data, public health studies often were limited at macro scale levels such as county level, and may not allow public health researchers and health officials to adequately identify, analyze, and monitor health problems at the community level. In this research [84], we take advantage of New York State SPARCS open dataset, which collects patient level detail on patient characteristics, diagnoses and treatments, services, and charges for each hospital inpatient stay and outpatient treatment. Such data also provides street level location information for each patient and healthcare facility site. Through geocoding and geo-mapping, we provide spatial oriented data analysis on New York state health records at the community level. We study geospatial distributions of diseases in New York State at multiple spatial resolutions, and provide multi-dimensional analysis by grouping patients into different groups. We discover potential spatial clusters, hot spots or anomalies of disease distributions. We will also study potential correlations between socio-economic determinants and diseases by integrating additional spatial datasets, including socio-economic data and environment data (air quality

indexes, pollen counts). Such large scale spatial oriented analytics will rely on scalable spatial querying and analytics methods at large scale.

Spatial data exploration includes density based spatial patterns such as spatial clusters, hotspots, and anomalies. In digital pathology, we will need to detect and quantify regions that are significant and different from others with high scores according to density measures and statistical testing of spatial objects. For example, for brain tumors studies, pseudopalisades appear as ring-enhancing lesions where the rings have much higher concentration of cells than adjacent regions. For public health studies, we use patients locations to discover spatial clusters of diseases and potential determinants associated with such clusters to monitor health problems at the community level.

There are major research challenges to support such spatial clustering at extreme scale due to the explosion of spatial data at microanatomic level and patient level. First, we will need to develop or adapt spatial clustering methods that can support detection of spatial clusters of complex shapes such as pseudopalisades. Second, we will need to make existing spatial clustering methods highly scalable for spatial big data.Stony Brook University will collaborate with Indiana University on developing scalable spatial clustering methods to support biomedical informatics problems driven by pathology imaging and GIS oriented public health studies.

5.5. Enabled Applications - Biomolecular Simulation Data Analysis

Overview of Biomolecular Simulation Data Analysis

Molecular dynamics (MD) simulations have become an important computational tool to study biomolecular systems [85-87], in particular membrane proteins and membrane system [88-91]. Analysis of molecular dynamics (MD) trajectories is becoming more and more challenging with simulation times routinely exceeding microseconds (with millions of frames) and increasing in size (with millions of particles). The increase in data volume is driven by (1) improvements in hardware (such as HPC systems with tens of thousands of cores and GPU accelerators) and algorithms [92-95] (2) use of multi-copy enhanced sampling methods [94, 96-99], and (3) new efficient representations of the physical interactions such as coarse-grained models, which allows simulations of larger systems and at longer time steps [100, 101]. Here we explore a number of challenging analysis tasks with the goal to establish a better understanding for the underlying problem classes and with a view towards prototyping SPIDAL based algorithms.

Hausdorff calculation for Path Similarity Analysis

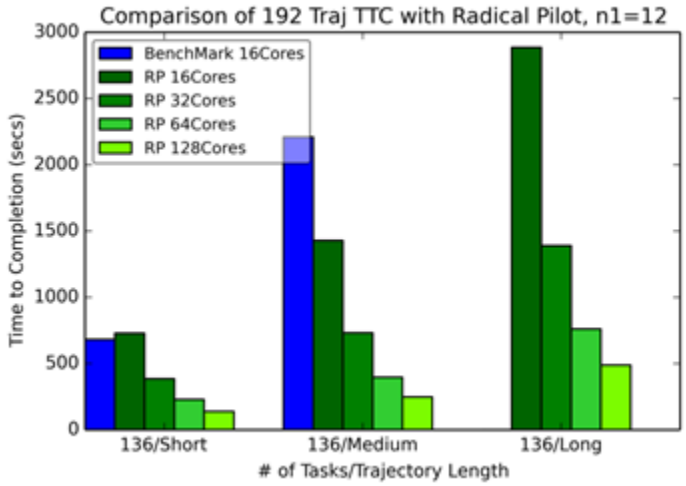


Figure 5-2: Time to Completion of a Short, Medium and Large trajectory PSA Analysis on XSEDE Stampede. As Benchmark is the time to completion by the PSA algorithm implementation in MDAnalysis.

Path Similarity Analysis (PSA) was introduced by Seyler et al.[102] in order to quantify the similarity between two arbitrary MD trajectories (which are considered geometrically as paths in the high-dimensional configuration space) and extract the atomic-scale determinants responsible for their differences. Given two trajectories with M_1 and M_2 frames, PSA uses the Hausdorff metric δ_{ij} [103] to compute a distance between two paths. The Hausdorff metric in turn requires the calculation of a distance d_{ij} between all frames $1 \leq i \leq M_1$ and $1 \leq j \leq M_2$. The distance function is typically a metric such as the Euclidean metric in the 3N-dimensional configuration space, where N is the number of atoms. In typical applications, N is on the order of 10^3 to 10^4, and M_1 and M_2 can be anywhere between 10^2 and 10^7. Furthermore, PSA is typically applied to an ensemble of trajectories, typically containing hundreds of trajectories ($n > 100$). The output of PSA is a matrix of Hausdorff distances between all trajectories. By clustering the distance matrix, similarity relationships between trajectories can be revealed [102].

At its core, the Hausdorff distance calculation for an ensemble of trajectories is an all-pairs problem. We implemented the PSA algorithm (done in MDAnalysis [102, 104]) into the RADICAL-Pilot framework by utilizing the "all pairs" execution pattern provided by Ensemble Toolkit [105]. The RADICAL-Pilot framework's implementation of the PSA algorithm calculates the distances in smaller independent segments. Each segment of calculations is executed as a single task. By employing the task level parallelism capabilities of the RADICAL-Pilot framework the PSA algorithm can be executed in an efficient and scalable manner. Benchmarks with three different trajectory sizes are shown in Figure 5-2.

The next step is to implement the Hausdorff distance algorithm with Yarn/Spark and understand the benefits of the in-memory execution that Spark provides. We will continue with the integration of the different PSA metrics provided by MDAnalysis. Generally, applications that have similar

properties partially or in total with the all-pairs problem can benefit from the RADICAL-Pilot framework approach.

Topological analysis of lipid membranes

Biological membranes are lipid bilayers with distinct inner and outer surfaces that are formed by lipid monolayers ("leaflets"). Movement of lipids between leaflets or changes in the membrane topology such as the merging of leaflets during a fusion event between two cells or vesicles is difficult to detect in simulations [106]. Understanding the underlying physics is important for biological transport processes in the synapses [107] and the Golgi apparatus [108, 109] but might also be of interest for the development of drug delivery vehicles [110].

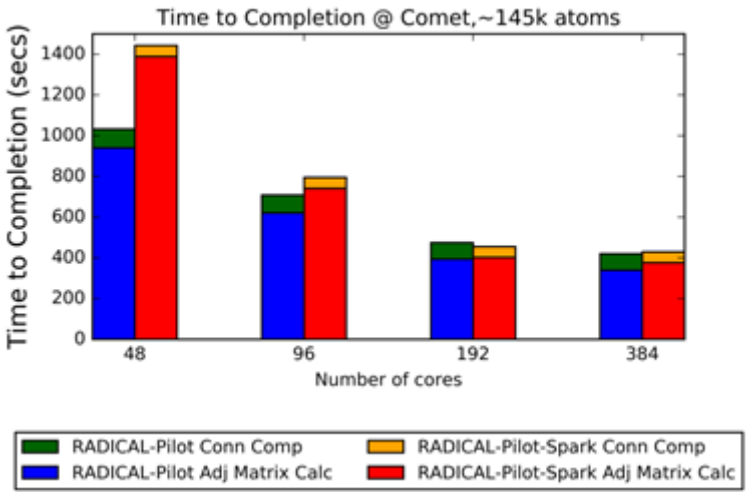


Figure 5-3: Time to completion of LeafletFinder algorithm with RADICA-Pilot and RADICAL-Pilot Spark on Comet for a system of 145000 atoms.

The LeafletFinder algorithm in MDAnalysis[104] is a graph-based algorithm that assigns individual lipids to topologically distinct leaflets. In short, the algorithm proceeds in two steps. In the first step, a nearest neighbor problem has to be solved in order to find lipid headgroups within a given cutoff. From the resulting adjacency matrix, a graph is constructed. In a second step, the largest connected subgraphs in the graph are found and sorted by size. With an appropriately chosen cutoff, each subgraph corresponds to a topologically distinct leaflet. The current implementation of LeafletFinder is slow for medium sized systems (> 1000 lipids). It is also an interesting test case because of two distinct algorithmic steps, which are likely to exhibit different scaling and optimization requirements. There are two different implementations of LeafletFinder in the RADICAL-Pilot framework. The first implementation utilized task level parallelism. The adjacency matrix is calculated using the "all-pairs" pattern by a fixed number of independent tasks concurrently. The second step of the algorithm is executed as a single task after the adjacency matrix is calculated. The second implementation used Pilot-Spark.

Both implementations, the task level parallel and Spark, were tested over a large system (with more than 100000 lipids). The results obtained in Figure 5-3, show that the execution of the

specific algorithm could scale and finish quite quickly. Interestingly, the Spark implementation was not faster than the vanilla, strengthening the opinion that the two algorithmic steps will have different optimization requirements. The test was executed on XSEDE Comet utilizing up to 384 cores (up to 16 nodes). Currently, we are investigating what type of optimizations are possible to do in the Spark implementation of the algorithm and what type of processing is needed in each step of the algorithm.

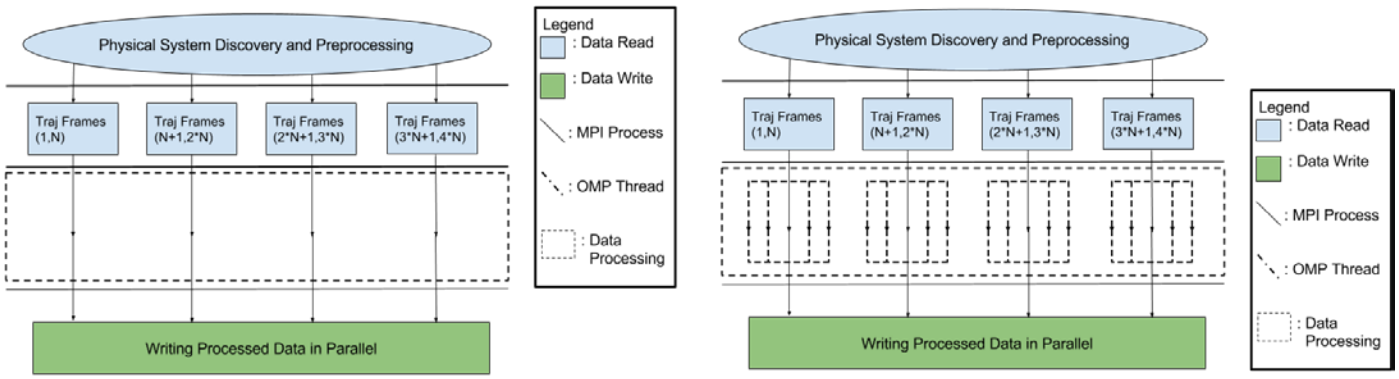


Figure 5-4: CPPTRAJ Levels of Parallelism

CPPTRAJ Amber Data Analysis

CPPTRAJ is the main program in the Amber molecular dynamics software package for processing and analyzing various data output from MD simulations. We have recently made several additions to the code which have greatly improved both the speed and the utility of CPPTRAJ.

Due to massive increases in processor power and the widespread use of enhanced sampling methods that generate ensembles of trajectories, it is now commonplace for tens to hundreds of GB of MD data to be generated in a single run. Analysis tools must be able to keep up with this deluge of data. For the past few years, CPPTRAJ has had the ability to handle large amounts of data via two levels of parallelism: processing of ensembles of trajectories with MPI, where each thread is responsible for processing a single trajectory in the ensemble (across-ensemble parallelism), and OpenMP-parallelization of time-consuming calculations. CPPTRAJ now has a third level of parallelism in which trajectory reads/writes can also be divided among MPI threads (across-trajectory parallelism). In addition, all three levels of parallelism can be active at the same time (hybrid MPI/OpenMP). For example, say you have a small high-performance computing cluster with 16 available nodes, and each node has 16 cores. Given an 8 trajectory ensemble, CPPTRAJ could make use of all available resources; the processing of each ensemble trajectory could be divided among 2 nodes each, and calculations on each frame of the trajectories could

utilize all 16 cores of each node. These enhancements allow us to better utilize resources of HPC clusters, as well as process extremely large data sets in a much shorter amount of time. In addition, the scaling of across-trajectory parallelism is quite good since no communication is required between threads during processing. We are also working with the Jha lab to enable use of the Radical Pilot framework for asynchronous data processing in addition to investigating the use of the Hadoop-like capabilities and SPIDAL tools.

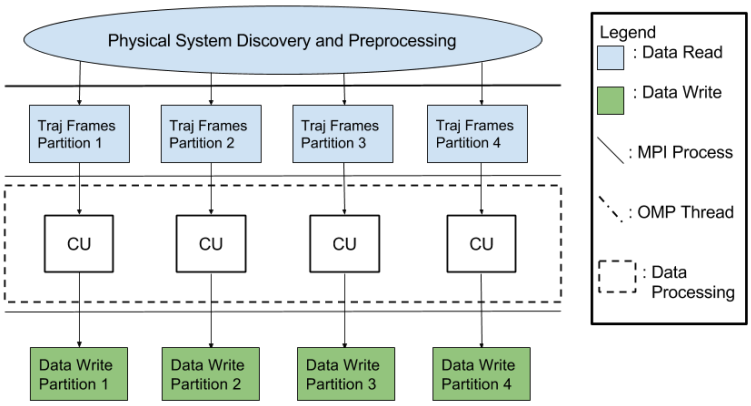


Figure 5-5: RADICAL-Pilot CPPTraj Analysis scenario

CPPTraj's levels of parallelism, as mentioned above, are depicted in Figure 5-4. CPPTraj offers a balanced across-trajectory division to the MPI threads. Employing the task level parallelism, offered by RADICAL-Pilot, we can achieve a more elaborate data partitioning. We are investigating the scenario (Figure 5-5) where the data are partitioned and each partition is now executed through a Compute Unit. The Compute Unit can be a CPPTraj executable that may or may not use the levels of parallelism already provided. Based on experimental results, over different types and sizes of data, we will be able to create a heuristic which will allow us to decide which method of analysis is better for a given dataset.

We have also been improving the speed of very time-consuming calculations via GPU acceleration. The 'closest' action, which retains a specific number of solvent molecules around a specified region of solute, typically requires a large amount of distance calculations. By offloading the distance calculations to the GPU (using CUDA), we have been able to obtain several orders of magnitude speedup over the single threaded code, and an order of magnitude speedup over the OpenMP code. In addition, the CUDA code can be used with MPI trajectory parallelization for even greater overall speedup.

We are looking to the future of MD data not just in terms of increasing trajectory sizes, but increasing system sizes as well. Typical system sizes are now tens of thousands of atoms, and it is not uncommon to see systems of hundreds of thousands of atoms or more. Recent improvements to topology file parsing and internal handling of topology data in CPPTRAJ have allowed us to successfully process a chromatin fiber system consisting of over 11.5 million atoms.

Finally, there have been some improvements made to the clustering code in CPPTRAJ as well (GitHub version). Users now have the option to prevent storing the pairwise cache in memory and instead calculate pairwise distances on the fly. While this option is slower, it does greatly increase the possible size of data sets that can be clustered. Users can also specify that the same pairwise distance matrix be used for several consecutive clustering instances. This means that pairwise distances only need to be calculated once, which can be quite useful for assessing clustering results using different input parameters (e.g. epsilon, number of clusters, etc) in a more time-efficient manner.

Much of this code was released in April 2016 with Amber16 and the remaining code has been made available in the GitHub version. New features are described at [111].

6 Community Engagement

6.1. REU Programs

6.2. Making MIDAS and SPIDAL Available to Community

6.3. Working with Apache: Harp and Heron

6.1. REU Programs

We have obtained supplements and offered REU programs for the SPIDAL institutions in the first two years and intend to continue this [112].

6.2. Making MIDAS and SPIDAL Available to the Community

SPIDAL-MIDAS: We will employ a three step approach to make SPIDAL-MIDAS developments available to the community. These activities span community engagement, deployment and possible integration.

- **Engagement:** BoF at SC'16 and SC'17 and Workshop/tutorial at XSEDE'17 and XSEDE'18.
- **Deployment:** Work with TACC, NCSA, SDSC, PSC to develop and deploy on XSEDE and Blue Waters resources.
- **Integration:** Work with other DIBBS and BIGDATA proposals, especially SDSC/ OSU led project "Scalable Middleware for Managing and Processing Big Data on Next Generation HPC Systems" [113].

Cloudmesh: We are expending effort to make Cloudmesh more accessible to the community. Cloudmesh has been available in the community as an open-source project and is available in Github [31] and pypi. In April 2016 it was downloaded 287 times. However, our biggest success is that Cloudmesh is now used and extended based on the needs of the NSF-sponsored Comet supercomputer. It also shows that Cloudmesh is uniquely positioned not only to support well established cloud frameworks, but also allows the support of state of the art academic virtual cluster efforts as brought forward by comet. In addition, we are presenting a paper and a tutorial showcasing our efforts at XSEDE2016.

All of the Cloudmesh software is available through open source. We include in [31] the list of relevant repositories hosted on Github or links to our software hosted elsewhere:

SPIDAL Examples: Along with SPIDAL's core algorithms comes a separate examples repository with code, data, and scripts to help a user get started quickly. These are available in [114]. The examples are designed to be able to deploy on a cluster or cloud VMs from scratch. Currently, these are based on Ubuntu – (16.04 or 14.04 preferred) Linux systems. To test on other systems such as RHEL or CentOS the scripts need to be slightly modified.

6.3. Working with Apache: Harp and Heron

The Apache Software Foundation (ASF) is a community-driven open source organization for hosting projects. To become a successful ASF member, it is important to build a vibrant community around your work. Most projects entering into Apache have previously been available as open source versions in places such as Github. Upon achieving a certain level of community support and publicity they can be introduced under the Apache umbrella for better visibility and a wider pool of contributors. This way when a project joins, it will have a substantial community and a process that it can use to build upon in the future.

For the Harp initiative, the first goal is to make it open to the public through Github. Then after attracting sufficient interest from the development community of HPC and Big Data Ecosystems, it can be moved to Apache. Becoming a successful open source project requires strategic partnerships between interested parties. Also a more community-driven development methodology has to be introduced. This means discussing the architectural and design issues in public, lowering the entry requirements for new developers, and making it easier for users to adopt the Harp software. After entering ASF, Harp can follow the Apache Incubation process to become a

top-level project or a sub-project under a larger effort like Hadoop, depending on the community requirements.

Twitter Heron is an open source distributed stream processing engine available in Github. It is in the early community building phase and will be introduced to Apache Software Foundation in the future. We are engaged with the Heron development team to enhance its capabilities to work seamlessly and efficiently in HPC environments. The improvements are directly going to the main development branch of Heron.

7 Futures

7.1. Integrating SPIDAL and MIDAS as Coherent Building Blocks

7.2. Orchestration and Workflow

7.3. Streaming

7.1. Integrating SPIDAL and MIDAS as Coherent Building Blocks

We have made significant progress in the overall structure of this project: Convergence Diamonds and Ogres, HPC-ABDS and the Architecture of Scalable Big Data Machine Learning Library. Further both MIDAS and SPIDAL have made good progress with several building blocks: SPIDAL Java, Harp, Pilot Jobs/Data and the over 20 scalable library members reported in Section 4. However, these building blocks do not exist in the form of a single integrated and coherent product. For example, there does not exist a SPIDAL library or a MIDAS middleware which either an application developer or an XSEDE resource provider can download. There is a non-trivial effort in integrating the building blocks so as to create a library or middleware product, which will provide a capability that is greater than the simple sum of the blocks. Single software engineering is required to integrate and package the building blocks. We have gathered further usability requirements from users of Apache libraries demonstrating that one needs carefully designed uniform programmatic and user interfaces. We are currently looking at approaches to this which could include using Apache ourselves as described in Section 6.3, but we need a pulse effort at the project level to push our software to the needed usability levels.

7.2. Orchestration and Workflow

We have been exploring Apache Beam linked to either Heron, Flink or Spark as a dataflow API. The universal data flow API provided by Beam can be translated to dataflow programs executed by Flink, Spark and Heron giving user the flexibility to quickly migrate the programs written using Beam API between different big data runtime environments. Furthermore the Beam API provides a Streaming and Batch API with the same constructs making the switch between streaming and batch processing seamless. Beam is an open source version of Google Cloud Dataflow. This is level 17 in HPC-ABDS. Orchestration is needed to link multiple SPIDAL and MIDAS components together and essential in many applications.

7.3. Streaming

Motivation: The analysis of data streaming from on-line instruments, large scale simulations, and distributed sensors now enables near real-time steering and control of complex systems such as scientific experiments, transportation systems, and urban environments. Bringing readily available, easy to program distributed streaming systems to HPC environments can help scientific discoveries in diverse application areas.

We are examining both modern streaming technology and streaming applications to see how they could use and extend the SPIDAL and MIDAS framework. For example, by considering the architecture and potential of the Heron Distributed Stream Processing Framework (DSPF), we identified some areas where we can improve Heron to support HPC applications in the spirit of HPC-ABDS

From our experiences we see some immediate requirements from streaming applications considering application areas of real time applications, parallel applications and large data applications. Here we summarize some of these requirements.

Real time applications

We explored this in a community workshop [115]. A real time application needs to process the data within a given QoS. The main requirements for these applications have to do with the scheduling of streaming tasks and communications among them. The areas of interest include [27, 116-119]

1. Use high performance interconnects(RDMA) for low latency high throughput data processing

2. Scheduling for guaranteed QoS.
3. Introduce efficient communication algorithms to further reduce the communication costs. These algorithms are especially relevant in collective communication operations.
4. Explore shared memory communications within nodes

Parallel applications

Some applications require parallel computations in order to reduce the processing times for a single stream of data. In fact, this is one approach to achieving real-time response when cloud computing tasks exceeds the capacity of a sequential processor to complete in the required time interval. Such a parallel computation requires synchronization and special communication APIs for achieving best performance. As in deep learning, GPU processing may be needed.

Large data processing applications

There is a class of scientific streaming applications needing to process very large data. For example these can be very large images in the size of Gigabytes. Support for such large data is challenging due to in-memory data processing adopted by the stream engines. We need mechanisms to process large files while keeping parts of the data in permanent storage. For different types of those applications described above, we can improve the data processing APIs of streaming engines and introduce application libraries and benchmarks.

Data processing APIs

Add support for scientific data types such as images occurring in astronomy and remote sensing. Explore integration with Apache Beam to support complete scientific workflows. The Java based APIs in Heron are suitable for data processing applications but can face difficulties in scientific application, especially when trying to integrate with already available libraries. The APIs can be implemented in C++ for fully integrating with HPC domain specific applications and libraries.

Application libraries & Benchmarks

Identify the common streaming applications for scientific communities and create libraries. Comprehensive set of benchmarks for streaming applications is useful for understanding the characteristics of streaming application performance.

8 Team & Publications

Indiana University

Geoffrey Fox
Professor of Informatics and Computing, and Physics

Judy Qiu
Associate Professor of Computer Science

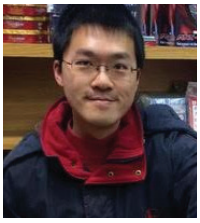
Gregor von Laszewski
Assistant Director, CGL and DSC, School of Informatics and Computing

David Crandall
Associate Professor, School of Informatics and Computing

Supun Kamburu-gamuve
PhD Candidate

Saliya Ekanayake
PhD Candidate

Pulasti Wick-ramasinghe
PhD Candidate



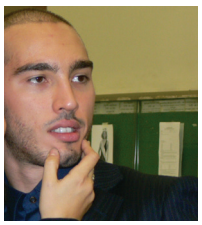
Bingjing Zhang
PhD Candidate

Not Pictured:
Bo Peng, Visiting Faculty
Mingze Xu, PhD Candidate

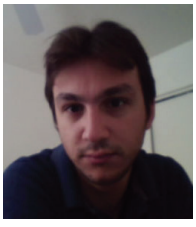
Rutgers University

Shantenu Jha
Associate Professor in ECE Conducts Research

Andre Luckow
Post-Doctoral Researcher, RADICAL Group



Alessio Angius
RADICAL Group Member



Ioannis Paraskevakos
PhD Student, RADICAL Group Member

Sean Olejar
Undergraduate Student under Dr. Jha

Virginia Tech

Madhav Marathe
Director, Network Dynamics and Simulation Science Lab, Biocomplexity Institute

Maleq Khan
Research Scientist at Virginia Bioinformatics Institute

Anil Kumar S. Vullikanti
Associate Professor, Dept. of Computer Science and Biocomplexity Institute

University of Kansas

John Paden
Associate Scientist, Center for Remote Sensing of Ice Sheets

Not Pictured:

Teresa Stumpf, MS EE 2015 (now at John Hopkins Applied Physics Lab)

Sean Holloway, REU 2015 (now MS EE at Columbia University)

Sravya Athinarapu, MS EE 2017

Jordan Sprick, REU 2016

Stony Brook University

Fusheng Wang
Assistant Professor Dept. of Biomedical Informatics and Dept. of Computer Science

Jun Kong
Assistant Professor, Emory University

Not Pictured:

Pengyue Zhang, PhD Student

Yanhui Liang, PhD Student

Furqan Baig, PhD Student

Hoang Vo, PhD Student

Xin Chen, PhD Student

Arizona State University

Oliver Beckstein
Leader, Computational Biophysics Research Group, Center for Biological Physics and Dept. of Physics

University of Utah

Not Pictured:

Thomas Cheatham, Professor of Medicinal Chemistry and Director of Research Computing

Daniel Roe

Rodrigo Galindo-Murillo

List of Publications

I. Saliya Ekanayake, Supun Kamburugamuve and Geoffrey Fox, "SPIDAL: High Performance Data Analytics with Java and MPI on Large Multicore HPC Clusters", Technical Report January 5 2016, Proceedings of 24th High Performance Computing Symposium (HPC 2016), April 3-6, 2016, Pasadena, CA, USA as part of the SCS Spring Simulation Multi-Conference (SpringSim'16).

II. Supun Kamburugamuve, Saliya Ekanayake, Milinda Pathirage, Geoffrey Fox, "Towards High Performance Processing of Streaming Data in Large Data Centers" Technical Report January 26 2016, to be published in proceedings of HPBDC 2016 IEEE International Workshop on High-Performance Big Data Computing in conjunction with The 30th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2016), Chicago Hyatt Regency, Chicago, Illinois USA, Friday, May 27th, 2016.

III. Bingjing Zhang, Peng Bo and Judy Qiu, "Model-Centric Computation Abstractions in Machine Learning Applications", Proceedings of the 3rd Workshop o Algorithms and Systems for MapReduce and Beyond (BeyondMR2016), held in conjunction with SIGMOD 2016, San Francisco, California, July 1, 2016.

IV. Bingjing Zhang, Bo Peng, Judy Qiu, High Performance LDA through Collective Model Communication Optimization, Proceedings of International Conference on Computational Science (ICCS2016) conference, San Diego, California, June 6-8, 2016.

V. Ablimit Aji and Fusheng Wang, Challenges and Approaches in Spatial Big Data Management. Big Data: Storage, Sharing, and Security (3S) Fei Hu. Auerbach Publications. 2016. ISBN: 978-1-4987-3486-8.

VI. Cong Xie, Wen Zhong, Jun Kong, Wei Xu, Klaus Mueller, and Fusheng Wang, IEVQ: An Iterative Example-based Visual Query for Pathology Database. Proceedings of the Second International Workshop on Data Management and Analytics for Medicine and Healthcare. 2016.

VII. Hoang Vo, Jun Kong, Dejun Teng, Yanhui Liang, Ablimit Aji, George Teodoro and Fusheng Wang. A MapReduce Based High Performance Whole Slide Image Analysis Framework in the Cloud. Proceedings of the Second International Workshop on Data Management and Analytics for Medicine and Healthcare. 2016.

VIII. Jun Kong, Pengyue Zhang, Yanhui Liang, George Teodorou, Daniel J. Brat and Fusheng Wang, Robust Cell Segmentation for Histological Images of Glioblastoma. International Symposium on Biomedical Imaging (ISBI 2016).

IX. Xin Chen and Fusheng Wang. Integrative Spatial Data Analytics for Public Health Studies of New York State. Proceedings of AMIA 2016 Annual Symposium.

X. Alam M, Khan M, Vullikanti A, Marathe M, An Efficient and Scalable Algorithmic Method for Generating Large–Scale Random Graphs. In proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Salt Lake City, UT, November 13-18, 2016.

XI. Andre Luckow, Ioannis Paraskevakos, George Chantzialexiou, Shantenu Jha, Hadoop on HPC: Integrating Hadoop and Pilot-based Dynamic Resource Management, Workshop on High-Performance Big Data Computing, 2016. [link] [bib]: hadoop-on-hpc http://arxiv.org/abs/1602.00345.

XII. Yanhui Liang, Jun Kong, Yangyang Zhu and Fusheng Wang, Three-Dimensional Data Analytics for Pathology Imaging. First International Workshop on Data Management and Analytics for Medicine and Healthcare (DMAH 2015).

XIII. Arifuzzaman S, Khan M, Marathe M, A Space-efficient Parallel Algorithm for Counting Exact Triangles in Massive Networks. In Proceedings of the 17th IEEE International Conference on High Performance Computing and Communications. New York City, NY, August 24-26, 2015.

XIV. Bing Zhang, Yang Ruan, and Judy Qiu, Harp: Collective Communication on Hadoop, Proceedings of IEEE International Conference on Cloud Computing Engineering (IC2E), Tempe, Arizona, March 9-12, 2015.

XV. Yanhui Liang, Fusheng Wang, Darren Treanor, Derek Magee, George Teodoro, Yangyang Zhu and Jun Kong, Whole-Slide Histological Image Analysis for 3D Primary Vessel Reconstruction. The 18th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2015). Munich, Germany.

9 References

[1] Geoffrey L. House, Saliya Ekanayake, Yang Ruan, Ursel Schütte, Wittaya Kaonongbua, Geoffrey Fox, Yuzhen Ye, and James D. Bever, Phylogenetically structured differences in rRNA gene sequence variation among species of arbuscular mycorrhizal fungi and their implications for sequence clustering", , , June 3 2016. Applied and Environmental Microbiology (American Society of Microbiology), June 3, 2016. DOI: http://doi.org/10.1128/AEM.00816-16

[2] Yang Ruan, Saliya Ekanayake, Mina Rho, Haixu Tang, Seung-Hee Bae, Judy Qiu, and Geoffrey Fox, DACIDR: deterministic annealed clustering with interpolative dimension reduction using a large collection of 16S rRNA sequences, in Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine. 2012, ACM. Orlando, Florida. pages. 329-336. http://grids.ucs.indiana.edu/ptliupages/publications/DACIDR_camera_ready_v0.3.pdf. DOI: 10.1145/2382936.2382978.

[3] Yang Ruan, G.L.H., Saliya Ekanayake, Ursel Schütte, James D. Bever, Haixu Tang, Geoffrey Fox, Integration of Clustering and Multidimensional Scaling to Determine Phylogenetic Trees as Spherical Phylograms Visualized in 3 Dimensions, in C4Bio 2014 of IEEE/ACM CCGrid 2014. May 26-29, 2014, 2014. Chicago, USA. http://salsahpc.indiana.edu/millionseq/fungi2_phylo/reference/Integration%20of%20Clustering%20and%20Multidimensional%20Scaling%20to%20Determine%20Phylogenetic%20Trees%20as%20Spherical%20Phylograms%20Visualized%20in%203%20Dimensions.pdf.

[4] Maksudul Alam and Maleq Khan, Parallel Algorithms for Generating Random Networks with Given Degree Sequences. International Journal of Parallel Programming,, 2015. http://arxiv.org/abs/1406.1215

[5] Shantenu Jha, Judy Qiu, Andre Luckow, Pradeep Mantha, and Geoffrey C.Fox, A Tale of Two Data-Intensive Paradigms: Applications, Abstractions, and Architectures, in Big Data (BigData Congress), 2014 IEEE International Congress on. June 27 2014-July 2 2014, 2014. pages. 645-652. http://arxiv.org/pdf/1403.1528. DOI: 10.1109/BigData.Congress.2014.137.

[6] Geoffrey Fox, Judy Qiu, Shantenu Jha, Supun Kamburugamuve, and Andre Luckow, HPC-ABDS High Performance Computing Enhanced Apache Big Data Stack, in Invited talk at 2nd International Workshop on Scalable Computing For Real-Time Big Data Applications (SCRAMBL'15) at.CCGrid2015, the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. 2015, IEEE. Shenzhen, Guangdong, China. http://dsc.soic.indiana.edu/publications/HPC-ABDSDescribedv2.pdf.

[7] HPC-ABDS Kaleidoscope of over 350 Apache Big Data Stack and HPC Tecnologies. [accessed 2016 January 20]; Available from: http://hpc-abds.org/kaleidoscope/.

[8] Geoffrey Fox, Judy Qiu, and Shantenu Jha, High Performance High Functionality Big Data Software Stack, in Big Data and Extreme-scale Computing (BDEC). 2014. Fukuoka, Japan. http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/fox.pdf.

[9] Geoffrey C.Fox, Shantenu Jha, Judy Qiu, and Andre Luckow, Ogres: A Systematic Approach to Big Data Benchmarks, in Big Data and Extreme-scale Computing (BDEC) January 29-30, 2015. Barcelona. http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/OgreFacets.pdf.

[10] Geoffrey C. Fox, Shantenu Jha, Judy Qiu, and Andre Luckow, Towards an Understanding of Facets and Exemplars of Big Data Applications, in 20 Years of Beowulf: Workshop to Honor Thomas Sterling's 65th Birthday April 13, 2015. Annapolis http://dsc.soic.indiana.edu/publications/OgrePaperv11.pdf. DOI: http://dx.doi.org/10.1145/2737909.2737912.

[11] Geoffrey C. FOX , Shantenu JHA, Judy QIU, Saliya EKANAYAKE, and Andre LUCKOW, Towards a Comprehensive Set of Big Data Benchmarks, Chapter in Big Data and High Performance Computing, Lucio Grandinetti and Gerhard Joubert, Editors. 2015, IOS. http://grids.ucs.indiana.edu/ptliupages/publications/OgreFacetsv9.pdf. DOI: 10.3233/978-1-61499-583-8-47.

[12] Geoffrey Fox, Judy Qiu, Shantenu Jha, Saliya Ekanayake, and Supun Kamburugamuve, Big Data, Simulations and HPC Convergence. January 30, 2016. http://dsc.soic.indiana.edu/publications/HPCBigDataConvergence.pdf. DOI: https://www.researchgate.net/publication/301231174_Big_Data_Simulations_and_HPC_Convergence.

[13] Geoffrey Fox, Judy Qiu, Shantenu Jha, Saliya Ekanayake, and Supun Kamburugamuve, White Paper: Big Data, Simulations and HPC Convergence, in BDEC Frankfurt workshop. June 16, 2016. Frankfurt Airport, Germany.

http://dsc.soic.indiana.edu/publications/HPCBigDataConvergence.Summary_IURutgers.pdf. DOI: http://dx.doi.org/10.13140/RG.2.1.3112.2800.

[14] Digital Science Center. SPIDAL Home Page: CIF21 DIBBs: Middleware and High Performance Analytics Libraries for Scalable Data Science - Scalable Parallel Interoperable Data Analytics Library. 2015 [accessed 2015 June 21]; Available from: http://spidal.org/index.html.

[15] Indiana University News Release: IU computer scientists receive $5 million to empower U.S. researchers with new data analysis tools. 2014 October 28 [accessed 2016 July 7]; Available from: http://news.indiana.edu/releases/iu/2014/10/big-data-dibbs-grant.shtml.

[16] NSF Award Announcement: Abstract #1443054: CIF21 DIBBs: Middleware and High Performance Analytics Libraries for Scalable Data Science. 2014 [accessed 2016 July 7]; Available from: http://www.nsf.gov/awardsearch/showAward?AWD_ID=1443054.

[17] Geoffrey Fox and Wo Chang, Big Data Use Cases and Requirements, in 1st Big Data Interoperability Framework Workshop: Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data March 18 - 21, 2014. San Diego Supercomputer Center, San Diego. http://grids.ucs.indiana.edu/ptliupages/publications/NISTUseCase.pdf.

[18] Judy Qiu, Shantenu Jha, Andre Luckow, and Geoffrey C.Fox, Towards HPC-ABDS: An Initial High-Performance Big Data Stack, in Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data. March 18-21, 2014. San Diego Supercomputer Center, San Diego. http://grids.ucs.indiana.edu/ptliupages/publications/nist-hpc-abds.pdf.

[19] Bingjing Zhang, Bo Peng, and Judy Qiu, Model-centric computation abstractions in machine learning applications, in Proceedings of the 3rd ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond. 2016, ACM. San Francisco, California. pages. 1-4. DOI: 10.1145/2926534.2926539.

[20] NIST Big Data Public Working Group: Use Cases and Requirements Subgroup, NIST Big Data Interoperability Framework: Volume 3, Use Cases and General Requirements (NIST Special Publication 1500-3). 2016, NIST: Vol. 3. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-3.pdf. DOI: http://dx.doi.org/10.6028/NIST.SP.1500-3.

[21] Krste Asanovic, Rastislav Bodik, James Demmel, Tony Keaveny, Kurt Keutzer, John Kubiatowicz, Nelson Morgan, David Patterson, Koushik Seny, John Wawrzynek, David Wessel, and Katherine Yelick, A View of Parallel Computing. Communications of the ACM, October, 2009. 52(10): p. 56-67. DOI:10.1145/1562764.1562783. http://portal.acm.org/citation.cfm?id=1562764.1562783&coll=portal&dl=ACM

[22] NASA Advanced Supercomputing Division. NAS Parallel Benchmarks. 1991 [accessed 2014 March 28]; Available from: https://www.nas.nasa.gov/publications/npb.html.

[23] R. F. Van der Wijngaart, S. Sridharan, and V. W. Lee. Extending the BT NAS Parallel Benchmark to exascale computing. in International Conference forHigh Performance Computing, Networking, Storage and Analysis (SC). 10-16 Nov. 2012.

[24] Committee on the Analysis of Massive Data; Committee on Applied and Theoretical Statistics; Board on Mathematical Sciences and Their Applications; Division on Engineering and Physical Sciences; National Research Council, Frontiers in Massive Data Analysis. 2013: National Academies Press. http://www.nap.edu/catalog.php?record_id=18374

[25] Bingjing Zhang, Bo Peng, and Judy Qiu, High Performance LDA through Collective Model Communication Optimization, in International Conference on Computational Science 6-8 June, 2016, Procedia Computer Science: Vol. 80. San Diego, California,. pages. 86-97. http://www.sciencedirect.com/science/article/pii/S1877050916306512. DOI: http://dx.doi.org/10.1016/j.procs.2016.05.300.

[26] Bingjing Zhang, Yang Ruan, and Judy Qiu, Harp: Collective Communication on Hadoop, in IEEE International Conference on Cloud Engineering (IC2E). March 9-12, 2015. Tempe AZ. http://grids.ucs.indiana.edu/ptliupages/publications/HarpQiuZhang.pdf.

[27] Supun Kamburugamuve, Saliya Ekanayake, Milinda Pathirage, and Geoffrey Fox, Towards High Performance Processing of Streaming Data in Large Data Centers, in HPBDC 2016 IEEE International Workshop on High-Performance Big Data Computing in conjunction with The 30th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2016). May 27, 2016. Chicago. http://dsc.soic.indiana.edu/publications/high_performance_processing_stream.pdf.

[28] Saliya Ekanayake, Supun Kamburugamuve, and Geoffrey Fox, SPIDAL: High Performance Data Analytics with Java and MPI on Large Multicore HPC Clusters, in 24th High Performance Computing Symposium (HPC 2016), , 2016, as part of the SCS Spring Simulation Multi-Conference (SpringSim'16). April 3-6, 2016. Pasadena, CA, USA http://dsc.soic.indiana.edu/publications/hpc2016-spidal-high-performance-submit-18-public.pdf.

[29] Java Grande Home Page. 2002 [accessed 2016 July 9]; Available from: http://www.javagrande.org/.

[30] OpenHFT Thread Affinity Library. [accessed 2016 July 10]; Available from: https://github.com/OpenHFT/Java-Thread-Affinity.

[31] Digital Science Center. cloudmesh/client: A light weight cloud client to manage virtual clusters. [accessed 2016 July 7]; Available from: http://cloudmesh.github.io/client.

[32] Rick Wagner, Philip Papadopoulos, Dmitry Mishin, Trevor Cooper, Mahidhar Tatineti, Gregor von Laszewski, Fugang Wang, and Geoffrey C. Fox, User Managed Virtual Clusters in Comet, in XSEDE 2016. July 17-21, 2016. Miami, FL. https://xsede16.sched.org/event/7RRV/tech-user-managed-virtual-clusters-in-comet.

[33] Andre Luckow, Ioannis Paraskevakos, George Chantzialexiou, and Shantenu Jha, Hadoop on HPC: Integrating Hadoop and Pilot-based Dynamic Resource Management in Workshop on High-Performance Big Data Computing, 2016 at IPDPS 2016. 2016, IEEE. Chicago, ILL. http://arxiv.org/abs/1602.00345.

[34] Andre Luckow, Mark Santcroos, Ashley Zebrowski, and Shantenu Jha, Pilot-Data. J. Parallel Distrib. Comput., 2015. 79(C): p. 16-30. DOI:10.1016/j.jpdc.2014.09.009

[35] Lise Getoor and Christopher P Diehl, Link mining: a survey. ACM SIGKDD Explorations Newsletter, 2005. 7(2): p. 3-12.

[36] Jure Leskovec, Ajit Singh, and Jon Kleinberg, Patterns of influence in a recommendation network, in Proceedings of the 10th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining. 2006, Springer-Verlag. Singapore. pages. 380-389. DOI: 10.1007/11731139_44.

[37] Xifeng Yan, X. Jasmine Zhou, and Jiawei Han, Mining closed relational graphs with connectivity constraints, in Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. 2005, ACM. Chicago, Illinois, USA. pages. 324-333. DOI: 10.1145/1081870.1081908.

[38] Eric Bloedorn, Neal J Rothleder, David DeBarr, and Lowell Rosen, Relational graph analysis with real-world constraints: An application in irs tax fraud detection, in The Twentieth National Conference on Artificial Intelligence (AAAI-05) July 9-13, 2005. Pittsburgh, Pennsylvania. http://www.aaai.org/Papers/Workshops/2005/WS-05-07/WS05-07-006.pdf.

[39] Milo, R., S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, Network motifs: simple building blocks of complex networks. Science, 2002. 298(5594): p. 824.

[40] Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, and George Karypis, Frequent Substructure-Based Approaches for Classifying Chemical Compounds. IEEE Trans. on Knowl. and Data Eng., 2005. 17(8): p. 1036-1050. DOI:10.1109/tkde.2005.127

[41] Zhao Zhao, Guanying Wang, Ali R. Butt, Maleq Khan, V. S. Anil Kumar, and Madhav V. Marathe, SAHAD: Subgraph Analysis in Massive Networks Using Hadoop, in Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium. 2012, IEEE Computer Society. pages. 390-401. DOI: 10.1109/ipdps.2012.44.

[42] Alon, N., R. Yuster, and U. Zwick, Color-coding. J. ACM, 1995. 42(4): p. 844-856. DOI:10.1145/210332.210337

[43] Intel. Intel® Data Analytics Acceleration Library. 2015 August 25 [accessed 2016 July 10]; Available from: https://software.intel.com/en-us/blogs/daal.

[44] Colfax Research. MCDRAM as High-Bandwidth Memory (HBM) in Knights Landing Processors: Developer's Guide. 2015 May 11 [accessed 2016 July 8]; Available from: http://colfaxresearch.com/knl-mcdram/.

[45] Joel C. Miller and Aric Hagberg, Efficient generation of networks with given expected degrees, in Proceedings of the 8th international conference on Algorithms and models for the web graph. 2011, Springer-Verlag. Atlanta, GA. pages. 115-126.

[46] Shumo Chu and James Cheng, Triangle listing in massive networks and its applications, in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. 2011, ACM. San Diego, California, USA. pages. 672-680. DOI: 10.1145/2020408.2020513.

[47] Miller McPherson, Lynn Smith-Lovin, and James M Cook, Birds of a Feather: Homophily in Social Networks. Annual Review of Sociology, 2001. 27(1): p. 415-444. DOI:doi:10.1146/annurev.soc.27.1.415. http://www.annualreviews.org/doi/abs/10.1146/annurev.soc.27.1.415

[48] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis, Efficient semi-streaming algorithms for local triangle counting in massive graphs, in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008, ACM. Las Vegas, Nevada, USA. pages. 16-24. DOI: 10.1145/1401890.1401898.

[49] Arnau Prat-Perez, David Dominguez-Sal, Josep-M. Brunat, and Josep-Lluis Larriba-Pey, Put Three and Three Together: Triangle-Driven Community Detection. ACM Trans. Knowl. Discov. Data, 2016. 10(3): p. 1-42. DOI:10.1145/2775108

[50] Shaikh Arifuzzaman, Maleq Khan, and Madhav Marathe, PATRIC: a parallel algorithm for counting triangles in massive networks, in Proceedings of the 22nd ACM international conference on Conference on information and knowledge management. 2013, ACM. San Francisco, California, USA. pages. 529-538. DOI: 10.1145/2505515.2505545.

[51] Jure Leskovec, Kevin J. Lang, and Michael Mahoney, Empirical comparison of algorithms for network community detection, in Proceedings of the 19th international conference on World wide web. 2010, ACM. Raleigh, North Carolina, USA. pages. 631-640. DOI: 10.1145/1772690.1772755.

[52] E. Jason Riedy, Henning Meyerhenke, David Ediger, and David A. Bader, Parallel community detection for massive graphs, in Proceedings of the 9th international conference on Parallel Processing and Applied Mathematics - Volume Part I. 2012, Springer-Verlag. Torun, Poland. pages. 286-296. DOI: 10.1007/978-3-642-31464-3_29.

[53] Yuzhou Zhang, Jianyong Wang, Yi Wang, and Lizhu Zhou, Parallel community detection on large networks with propinquity dynamics, in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. 2009, ACM. Paris, France. pages. 997-1006. DOI: 10.1145/1557019.1557127.

[54] J. Soman and A. Narang. Fast Community Detection Algorithm with GPUs and Multicore Architectures. in Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International. 16-20 May 2011 2011.

[55] Michael Ovelgönne, Distributed community detection in web-scale networks, in Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. 2013, ACM. Niagara, Ontario, Canada. pages. 66-73. DOI: 10.1145/2492517.2492518.

[56] Saliya Ekanayake. Global Machine Learning with DSC-SPIDAL. 2016 [accessed 2016 July 9]; Available from: https://www.gitbook.com/book/esaliya/global-machine-learning-with-dsc-spidal/details.

[57] Geoffrey Fox, D. R. Mani, and Saumyadipta Pyne, Parallel Deterministic Annealing Clustering and its Application to LC-MS Data Analysis, in IEEE International Conference on Big Data. October 6-9, 2013. Santa Clara, CA, USA, . DOI: http://dx.doi.org/10.1109/BigData.2013.6691636.

[58] Supun Kamburugamuve, Pulasthi Wickramasinghe, Saliya Ekanayake, Chathuri Wimalasena, Milinda Pathirage, and Geoffrey Fox, TSmap3D: Browser Visualization of High Dimensional Time Series Data. May 10, 2016. http://dsc.soic.indiana.edu/publications/tsmap3d.pdf.

[59] Yang Ruan PhD Thesis SCALABLE AND ROBUST CLUSTERING AND VISUALIZATION FOR LARGE-SCALE BIOINFORMATICS DATA, in School of Informatics and Computing. 18 August, Indiana University. http://dsc.soic.indiana.edu/publications/Final_Thesis_v1.03.pdf

[60] Geoffrey Fox, Robust Scalable Visualized Clustering in Vector and non Vector Semimetric Spaces. Parallel Processing Letters, June, 2013. 23(2). DOI:http://www.worldscientific.com/doi/abs/10.1142/S0129626413400069. http://grids.ucs.indiana.edu/ptliupages/publications/Clusteringv1.pdf

[61] Ekanayake, S., Y. Ruan, and G.C. Fox. Million Sequence Clustering. Available from: http://salsahpc.indiana.edu/millionseq/.

[62] Theresa M. Stumpf,Thesis A Wideband Direction of Arrival Technique for Multibeam, Wide-Swath Imaging of Ice Sheet Basal Morphology. , in M.S. Thesis, Electrical and Computer Science Department, 2015, University of Kansas

[63] David J. Crandall, Geoffrey C. Fox, John D. Paden, and Layer-finding in Radar Echograms using Probabilistic Graphical Models, in Technical Report submitted for publication. April 8, 2012. http://grids.ucs.indiana.edu/ptliupages/publications/icpr12-ice.pdf

[64] Stefan Lee, Jerome Mitchell, David J Crandall, and Geoffrey C Fox. Estimating bedrock and surface layer boundaries and confidence intervals in ice sheet radar imagery using MCMC. in 2014 IEEE International Conference on Image Processing (ICIP) 2014: IEEE.

[65] Daphne Koller and Nir Friedman, Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. 2009: The MIT Press. p.1208. ISBN:0262013193, 9780262013192

[66] Jun Kong , Lee A. D. Cooper, Fusheng Wang, Jingjing Gao, George Teodoro, Lisa Scarpace, Tom Mikkelsen, Matthew J. Schniederjan, Carlos S. Moreno, Joel H. Saltz, and Daniel J. Brat, Machine-Based Morphologic Analysis of Glioblastoma Using Whole-Slide Pathology Images Uncovers Clinically Relevant Molecular Correlates. PloS one, 2013. 8(11): p. e81049. DOI:http://dx.doi.org/10.1371/journal.pone.0081049

[67] Yanhui Liang, Fusheng Wang, Darren Treanor, Derek Magee, George Teodoro, Yangyang Zhu, and Jun Kong, A 3D Primary Vessel Reconstruction Framework with Serial Microscopy Images, Chapter in Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III, Nassir Navab, Joachim Hornegger, M. William Wells, and F. Alejandro Frangi, Editors. 2015, Springer International Publishing: Cham. p. 251-259. http://dx.doi.org/10.1007/978-3-319-24574-4_30. DOI: 10.1007/978-3-319-24574-4_30.

[68] Y. Liang, F. Wang, D. Treanor, D. Magee, G. Teodoro, Y. Zhu, and J. Kong. Liver whole slide image analysis for 3D vessel reconstruction. in 2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI). 16-19 April 2015 2015.

[69] Nicholas Roberts, Derek Magee, Yi Song, Keeran Brabazon, Mike Shires, Doreen Crellin, Nicolas M. Orsi, Richard Quirke, Philip Quirke, and Darren Treanor, Toward Routine Use of 3D Histopathology as a Research Tool. The American Journal of Pathology, 2012. 180(5): p. 1835-1842. DOI:10.1016/j.ajpath.2012.01.033. http://dx.doi.org/10.1016/j.ajpath.2012.01.033

[70] Yanhui Liang, Jun Kong, Yangyang Zhu, and Fusheng Wang, Three-Dimensional Data Analytics for Pathology Imaging, Chapter in Biomedical Data Management and Graph Online Querying: VLDB 2015 Workshops, Big-O(Q) and DMAH, Waikoloa, HI, USA, August 31 – September 4, 2015, Revised Selected Papers, Fusheng Wang, Gang Luo, Chunhua Weng, Arijit Khan, Prasenjit Mitra, and Cong Yu, Editors. 2016, Springer International Publishing: Cham. p. 109-125. http://dx.doi.org/10.1007/978-3-319-41576-5_8. DOI: 10.1007/978-3-319-41576-5_8.

[71] J. Kong, P. Zhang, Y. Liang, G. Teodoro, D. J. Brat, and F. Wang. Robust cell segmentation for histological images of Glioblastoma. in 2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI). 13-16 April 2016 2016.

[72] Furqan Baig, Mudit Mehrotra, Hoang Vo, Fusheng Wang, Joel Saltz, and Tahsin Kurc, SparkGIS: Efficient Comparison and Evaluation of Algorithm Results in Tissue Image Analysis Studies, Chapter in Biomedical Data Management and Graph Online Querying: VLDB 2015 Workshops, Big-O(Q) and DMAH, Waikoloa, HI, USA, August 31 – September 4, 2015, Revised Selected Papers, Fusheng Wang, Gang Luo, Chunhua Weng, Arijit Khan, Prasenjit Mitra, and Cong Yu, Editors. 2016, Springer International Publishing: Cham. p. 134-146. http://dx.doi.org/10.1007/978-3-319-41576-5_10. DOI: 10.1007/978-3-319-41576-5_10.

[73] CINET Cyberinfrastructure middleware to support Network Science. [accessed 2016 July 10]; Available from: http://cinet.vbi.vt.edu/granite/granite.html#login.

[74] Sherif Abdelhamid, Maksudul Alam, Richard Alo, Shaikh Arifuzzaman, Pete Beckman, Tirtha Bhattacharjee, Hasanuzzaman Bhuiyan, Keith Bisset, Stephen Eubank, Albert C. Esterline, Edward A. Fox, Geoffrey C. Fox, S. M. Shamimul Hasan, Harshal Hayatnagarkar, Maleq Khan, Chris J. Kuhlman, Madhav V. Marathe, Natarajan Meghanathan, Henning S. Mortveit, Judy Qiu, S. S. Ravi, Zalia Shams, Ongard Sirisaengtaksin, Samarth Swarup, Anil Kumar S. Vullikanti, and Tak-Lon Wu, CINET 2.0: A CyberInfrastructure for Network Science, in Proceedings of the 2014 IEEE 10th International Conference on e-Science - Volume 01. 2014, IEEE Computer Society. pages. 324-331. DOI: 10.1109/eScience.2014.21.

[75] Davis CA, C.G., Aiello LM, Chung K, Conover MD, Ferrara E, Flammini A, Fox GC, Gao X, Gonçalves B, Grabowicz PA, Hong K, Hui P, McCaulay S, McKelvey K, Meiss MR, Patil S, Peli Kankanamalage C, Pentchev V, Qiu J, Ratkiewicz J, Rudnick A, Serrette B, Shiralkar P, Varol

O, Weng L, Wu T, Younge AJ, Menczer F,, OSoMe: The IUNI observatory on social media. May 3, 2016, PeerJ Preprints. https://peerj.com/preprints/2008. DOI: https://doi.org/10.7287/peerj.preprints.2008v1.

[76] Xiaoming Gao, Evan Roth, Karissa McKelvey, Clayton Davis, Andrew Younge, Emilio Ferrara, Filippo Menczer, and Judy Qiu, Supporting a Social Media Observatory with Customizable Index Structures: Architecture and Performance, Chapter in Cloud Computing for Data-Intensive Applications, X. Li and J. Qiu, Editors. 2014, Springer New York: New York, NY. p. 401-427. http://dx.doi.org/10.1007/978-1-4939-1905-5_17. DOI: 10.1007/978-1-4939-1905-5_17.

[77] Pedro F. Felzenszwalb and Daniel P. Huttenlocher, Efficient Graph-Based Image Segmentation. Int. J. Comput. Vision, 2004. 59(2): p. 167-181. DOI:10.1023/b:visi.0000022288.19776.77

[78] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, DeepFace: Closing the Gap to Human-Level Performance in Face Verification, in Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. 2014, IEEE Computer Society. pages. 1701-1708. DOI: 10.1109/cvpr.2014.220.

[79] Andrej Karpathy, Armand Joulin, and Li Fei Fei. Deep fragment embeddings for bidirectional image sentence mapping. in Advances in Neural Information Processing Systems 2014.

[80] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, Microsoft COCO: Common Objects in Context, Chapter in Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, Editors. 2014, Springer International Publishing: Cham. p. 740-755. http://dx.doi.org/10.1007/978-3-319-10602-1_48. DOI: 10.1007/978-3-319-10602-1_48.

[81] Thorsten Joachims, Making large-scale support vector machine learning practical, Chapter in Advances in kernel methods, Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, Editors. 1999, MIT Press. p. 169-184.

[82] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in NIPS 2012, Advances in Neural Information Processing Systems 25. December 3-8, 2012, Curran Associates, Inc. Lake Tahoe. pages. 1097--1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[83] Ana Richelia Jara-Lazaro, Thomas Paulraj Thamboo, Ming Teh, and Puay Hoon Tan, Digital pathology: exploring its applications in diagnostic surgical pathology practice. Pathology, 2010. 42(6): p. 512-518.

[84] Xin Chen and Fusheng Wang, Integrative Spatial Data Analytics for Public Health Studies of New York State. , in AMIA 2016 Annual Symposium. 2016, American Medical Informatics Association. Chicago, ILL. http://www3.cs.stonybrook.edu/~fuswang/publications.html.

[85] Ron O. Dror, Robert M. Dirks, J.P. Grossman, Huafeng Xu, and David E. Shaw, Biomolecular Simulation: A Computational Microscope for Molecular Biology. Annual Review of Biophysics, 2012. 41(1): p. 429-452. DOI:doi:10.1146/annurev-biophys-042910-155245. http://www.annualreviews.org/doi/abs/10.1146/annurev-biophys-042910-155245

[86] Modesto Orozco, A theoretical view of protein dynamics. Chemical Society Reviews, 2014. 43(14): p. 5051-5066. DOI:10.1039/C3CS60474H. http://dx.doi.org/10.1039/C3CS60474H

[87] Juan R. Perilla, Boon Chong Goh, C. Keith Cassidy, Bo Liu, Rafael C. Bernardi, Till Rudack, Hang Yu, Zhe Wu, and Klaus Schulten, Molecular dynamics simulations of large macromolecular complexes. Current Opinion in Structural Biology, 4//, 2015. 31: p. 64-74. DOI:http://dx.doi.org/10.1016/j.sbi.2015.03.007. http://www.sciencedirect.com/science/article/pii/S0959440X15000342

[88] Siewert J. Marrink, Alex H. de Vries, and D. Peter Tieleman, Lipids on the move: Simulations of membrane pores, domains, stalks and curves. Biochimica et Biophysica Acta (BBA) - Biomembranes, 1//, 2009. 1788(1): p. 149-168. DOI:http://dx.doi.org/10.1016/j.bbamem.2008.10.006. http://www.sciencedirect.com/science/article/pii/S0005273608003325

[89] Phillip J Stansfeld and Mark S P. Sansom, Molecular Simulation Approaches to Membrane Proteins. Structure, 11/9/, 2011. 19(11): p. 1562-1572. DOI:http://dx.doi.org/10.1016/j.str.2011.10.002. http://www.sciencedirect.com/science/article/pii/S0969212611003364

[90] Julia Koehler Leman, Martin B. Ulmschneider, and Jeffrey J. Gray, Computational modeling of membrane proteins. Proteins, 11/19, 2015. 83(1): p. 1-24. DOI:10.1002/prot.24703. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4270820/

[91] Jing Li, Po-Chao Wen, Mahmoud Moradi, and Emad Tajkhorshid, Computational characterization of structural dynamics underlying function in active membrane transporters. Current Opinion in Structural Biology, 4//, 2015. 31: p. 96-105. DOI:http://dx.doi.org/10.1016/j.sbi.2015.04.001. http://www.sciencedirect.com/science/article/pii/S0959440X15000500

[92] Thomas E. Cheatham III and Daniel R. Roe, The Impact of Heterogeneous Computing on Workflows for Biomolecular Simulation and Analysis. Computing in Science & Engineering, 2015. 17(2): p. 30-39. DOI:doi:http://dx.doi.org/10.1109/MCSE.2015.7. http://scitation.aip.org/content/aip/journal/cise/17/2/10.1109/MCSE.2015.7

[93] David E. Shaw, Ron O. Dror, John K. Salmon, J. P. Grossman, Kenneth M. Mackenzie, Joseph A. Bank, Cliff Young, Martin M. Deneroff, Brannon Batson, Kevin J. Bowers, Edmond Chow, Michael P. Eastwood, Douglas J. Ierardi, John L. Klepeis, Jeffrey S. Kuskin, Richard H. Larson, Kresten Lindorff-Larsen, Paul Maragakis, Mark A. Moraes, Stefano Piana, Yibing Shan, and Brian Towles, Millisecond-scale molecular dynamics simulations on Anton, in Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. 2009, ACM. Portland, Oregon. pages. 1-11. DOI: 10.1145/1654059.1654099.

[94] Levi C. T. Pierce, Romelia Salomon-Ferrer, Cesar Augusto F. de Oliveira, J. Andrew McCammon, and Ross C. Walker, Routine Access to Millisecond Time Scale Events with Accelerated Molecular Dynamics. Journal of Chemical Theory and Computation, 2012/09/11, 2012. 8(9): p. 2997-3002. DOI:10.1021/ct300284c. http://dx.doi.org/10.1021/ct300284c

[95] Romelia Salomon-Ferrer, Andreas W. Götz, Duncan Poole, Scott Le Grand, and Ross C. Walker, Routine Microsecond Molecular Dynamics Simulations with AMBER on GPUs. 2. Explicit Solvent Particle Mesh Ewald. Journal of Chemical Theory and Computation, 2013/09/10, 2013. 9(9): p. 3878-3888. DOI:10.1021/ct400314y. http://dx.doi.org/10.1021/ct400314y

[96] Matthew C. Zwier and Lillian T. Chong, Reaching biological timescales with all-atom molecular dynamics simulations. Current Opinion in Pharmacology, 12//, 2010. 10(6): p. 745-752. DOI:http://dx.doi.org/10.1016/j.coph.2010.09.008. http://www.sciencedirect.com/science/article/pii/S1471489210001463

[97] Ayori Mitsutake, Yoshiharu Mori, and Yuko Okamoto, Enhanced Sampling Algorithms, Chapter in Biomolecular Simulations: Methods and Protocols, L. Monticelli and E. Salonen, Editors. 2013, Humana Press: Totowa, NJ. p. 153-195. http://dx.doi.org/10.1007/978-1-62703-017-5_7. DOI: 10.1007/978-1-62703-017-5_7.

[98] Sean L. Seyler and Oliver Beckstein, Sampling large conformational transitions: adenylate kinase as a testing ground. Molecular Simulation, 2014/08/09, 2014. 40(10-11): p. 855-877. DOI:10.1080/08927022.2014.919497. http://dx.doi.org/10.1080/08927022.2014.919497

[99] Rafael C. Bernardi, Marcelo C. R. Melo, and Klaus Schulten, Enhanced sampling techniques in molecular dynamics simulations of biological systems. Biochimica et Biophysica Acta (BBA) - General Subjects, 5//, 2015. 1850(5): p. 872-877.

DOI:http://dx.doi.org/10.1016/j.bbagen.2014.10.019. http://www.sciencedirect.com/science/article/pii/S0304416514003559

[100] Siewert J. Marrink and D. Peter Tieleman, Perspective on the Martini model. Chemical Society Reviews, 2013. 42(16): p. 6801-6822. DOI:10.1039/C3CS60093A. http://dx.doi.org/10.1039/C3CS60093A

[101] W. G. Noid, Perspective: Coarse-grained models for biomolecular systems. The Journal of Chemical Physics, 2013. 139(9): p. 090901. DOI:doi:http://dx.doi.org/10.1063/1.4818908. http://scitation.aip.org/content/aip/journal/jcp/139/9/10.1063/1.4818908

[102] Sean L Seyler, Avishek Kumar, Michael F Thorpe, and Oliver Beckstein, Path Similarity Analysis: A Method for Quantifying Macromolecular Pathways. PLoS Comput Biol, 2015. 11(10): p. e1004568.

[103] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge, Comparing Images Using the Hausdorff Distance. IEEE Trans. Pattern Anal. Mach. Intell., 1993. 15(9): p. 850-863. DOI:10.1109/34.232073

[104] Naveen Michaud-Agrawal, Elizabeth J. Denning, Thomas B. Woolf, and Oliver Beckstein, MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. Journal of Computational Chemistry, 2011. 32(10): p. 2319-2327. DOI:10.1002/jcc.21787. http://dx.doi.org/10.1002/jcc.21787

[105] Andre Luckow, Mark Santcroos, Ashley Zebrowski, and Shantenu Jha, Pilot-Data: An abstraction for distributed data. Journal of Parallel and Distributed Computing, 5//, 2015. 79–80: p. 16-30. DOI:http://dx.doi.org/10.1016/j.jpdc.2014.09.009. http://www.sciencedirect.com/science/article/pii/S0743731514001725

[106] Svetlana Baoukina and D. Peter Tieleman, Direct Simulation of Protein-Mediated Vesicle Fusion: Lung Surfactant Protein B. Biophysical Journal, 10/6/, 2010. 99(7): p. 2134-2142. DOI:http://dx.doi.org/10.1016/j.bpj.2010.07.049. http://www.sciencedirect.com/science/article/pii/S0006349510009215

[107] Thomas C Südhof, The synaptic vesicle cycle. Annu. Rev. Neurosci., 2004. 27: p. 509-547.

[108] Juan S. Bonifacino and Benjamin S. Glick, The Mechanisms of Vesicle Budding and Fusion. Cell, 1/23/, 2004. 116(2): p. 153-166. DOI:http://dx.doi.org/10.1016/S0092-8674(03)01079-1. http://www.sciencedirect.com/science/article/pii/S0092867403010791

[109] Graça Raposo and Willem Stoorvogel, Extracellular vesicles: Exosomes, microvesicles, and friends. The Journal of Cell Biology, February 18, 2013, 2013. 200(4): p. 373-383. DOI:10.1083/jcb.201211138. http://jcb.rupress.org/content/200/4/373.abstract

[110] Lisa M Bareford and Peter W Swaan, Endocytic mechanisms for targeted drug delivery. Advanced drug delivery reviews, 2007. 59(8): p. 748-758.

[111] CPPTRAJ wiki. [accessed 2016 July 8]; Available from: https://github.com/Amber-MD/cpptraj/wiki.

[112] REU Programs at Digital Science CEnter and SPIDAL Collaborators 2015. [accessed 2016 July 10]; Available from: http://www.dsc.soic.indiana.edu/reu2015.

[113] NSF Award Abstract #1447861: BIGDATA: F: DKM: Collaborative Research: Scalable Middleware for Managing and Processing Big Data on Next Generation HPC Systems. 2014 [accessed 2016 Jul 14]; Available from: http://nsf.gov/awardsearch/showAward?AWD_ID=1447861.

[114] SPIDAL Examples. 2002 [accessed 2016 July 12]; Available from: http://dsc-spidal.github.io/examples/.

[115] Geoffrey Fox, Lavanya Ramakrishnan, and Shantenu Jha, STREAM 2015 Final Report, in Streaming and Steering Applications: Requirements and Infrastructure October 27-28, 2015. Indianapolis, IN. DOI: http://streamingsystems.org/stream2015finalreport.html.

[116] Supun Kamburugamuve, Leif Christiansen, and Geoffrey Fox, A Framework for Real Time Processing of Sensor Data in the Cloud. Journal of Sensors, 2015. 2015: p. 11. DOI:10.1155/2015/468047. http://dsc.soic.indiana.edu/publications/iotcloud_hindavi_revised.pdf

[117] Supun Kamburugamuve, Hengjing He, Geoffrey Fox, and David Crandall, Cloud-based Parallel Implementation of SLAM for Mobile Robots, in International Conference on Internet of things and Cloud Computing (ICC 2016). March 22-23 2016. Cambridge, UK. http://dsc.soic.indiana.edu/publications/slam_isc_1.pdf.

[118] Supun Kamburugamuve and Geoffrey Fox, Survey of Distributed Stream Processing. January 8, 2016. http://dsc.soic.indiana.edu/publications/survey_distributed_stream_frameworks.pdf.

[119] Hengjing He, Supun Kamburugamuve, and G.C. Fox, Cloud based real-time multi-robot collision avoidance for swarm robotics. International Journal of Grid and Distributed Computing, May 7, 2015. http://dsc.soic.indiana.edu/publications/Cloud%20based%20real-time%20multi-robot%20collision%20avoidance%20for%20swarm%20robotics.pdf