Evaluation of SMP Shared Memory Machines for Use With In-Memory and OpenMP Big Data Applications

Andrew J. Younge*, Christopher Reidy†, Robert Henschel*, Geoffrey C. Fox*
*School of Informatics and Computing
Indiana University
901 E. 10th St, Bloomington, IN 47408, U.S.A.
{ajyounge, henschel, gcf}@indiana.edu
†Research Computing
University of Arizona
1077 N. Highland Ave, Tucson, AZ 85721, U.S.A.
chrisreidy@email.arizona.edu

***Abstract*—While distributed memory systems have shaped the field of distributed systems for decades, the demand for many-core shared memory resources is increasing. Symmetric Multiprocessor Systems (SMPs) have become increasingly important recently among a wide array of disciplines, ranging from Bioinformatics to astrophysics, and beyond. With the increase in big data computing, the size and scope of traditional commodity server systems is often outpaced. While some big data applications can be mapped to distributed memory systems found through many cluster and cloud technologies today, this effort represents a large barrier of entry that some projects cannot cross. Shared memory SMP systems look to effectively and efficiently fill this niche within distributed systems by providing high throughput and performance with minimized development effort, as the computing environment often represents what many researchers are already familiar with.**

In this paper we look at the use of two common shared memory systems, the ScaleMP vSMP virtualized SMP deployment at Indiana University, and the SGI UV architecture deployed at University of Arizona. While both systems are notably different in their design, their potential impact on computing are remarkably similar. As such, we look to compare each system first under a set of OpenMP threaded benchmarks via the SPEC group, and follow up with our experience using each machine for Trinity de-novo assembly. We find both SMP systems are well suited to support various big data applications, with the newer vSMP deployment often slightly faster, however certain caveats and performance considerations are necessary when considering such SMP systems.

***Index Terms*—Symmetric Multiprocessing, SMP, ScaleMP, vSMP, SGI UV, Large Memory, Big Data, Virtualization**

I. Introduction

Since the advent of the personal computer, industry and academia both have desired a bigger, more powerful single computing environment for a wide array of computational requirements. This can be seen most obviously in Moore's Law and the ever-increasing transistor count, but also the upward trend in main memory availability. However, there are natural limits to how large individual systems can grow, both in size and complexity. Today's latest Haswell CPUs can support 8 x 32GB DIMMS, or 256GB per socket. While this is an impressive feat achieved by hardware manufacturers, the reality is the computational tasks at hand today can require far more computational power that is only found when using distributed systems.

In the past few decades, distributed memory computing systems have offered the best solution for the growing computational requirements. Distributed memory systems involve building clusters, supercomputers, grids, and clouds [1] that piece together many individual systems together with high speed, low latency interconnects. The Top500 list [2] illustrates how almost all of the fastest computing systems today are clusters and supercomputers. However, using distributed shared memory systems often involve pushing the complexity of the architecture and challenges with parallelism directly to the software developers. While this is clearly not an insurmountable feat for some efforts, developing applications on distributed memory systems do represent a significant barrier of entry to parallel computing that can be challenging for smaller or niche projects to overcome.

In practice, there is the desire to treat a set of distributed resources as if it was a single, large entity, and leave the complexity of task orchestration to the system itself. Symmetric Multi-Processing systems, or SMP, represents the realization for this desire, whereby there is centralized shared memory under a single unified operating system with two or more processing units [3]. While today's SMP machines are actually multiple distributed systems with some layer of abstraction, they can be used as a single compute resource similar to a giant server. This usage model fits well with many data scientists who aren't used to or familiar with utilizing distributed architectures.

With the advent of big data within distributed systems and high performance computing [4], many new computing challenges are coming to the field. However, many efforts are not able to take advantage of distributed memory systems, for a number of reasons. This could be because the code set is outdated, the application is proprietary, the source code itself is unavailable, or the necessary technical skill is not available to the research community. Perhaps the large amount of development time it would take to rewrite an application far exceeds the application's potential utility within a given project. We expect SMP systems to fill this gap by providing a many-core and large memory system across a single unified OS, allowing for many big data applications to run either unmodified or with only minor code changes. However, the ability for these SMP systems to handle such big data workloads remains relatively unknown.

The Ligra project [5] is a great exmaple of how SMP machines can provide enough computational resources to complete a given task, without having to burden or over-complicate the design by utilizing distributed memory programming environments. Here, parallel graph processing algorithms are built to run in a shared memory environment, which greatly simplifies the design and implementation. The largest public graph datasets used within Ligra, a 127 billion edge synthetic graph, is able to fit into many of the SMP offerings. Furthermore, the Ligra project reports near-linear speedup using commodity threading techniques, illustrating how a parallel distributed memory implementation may not be necessary.

This paper is constructed as follows: First, explore the usage and variance in many-core SMP systems. Next, we show two different deployments at Indiana University and University of Arizona that represent two common but different SMP systems today. Then, we will look at evaluating each system in two different ways. First we will evaluate thread level parallelism with the SPEC OpenMP benchmark suite, and experiment with running Trinity, a de-novo genome assembly application that requires a large memory addressing space. Finally, we will conclude with a discussion of each machine and the relevance to shared memory systems within the computing community.

II. SMP Machines

While the use cases for SMP machines can vary greatly, today there generally exists two major use cases for SMPs.

1) *Thread Level Parallelism*- Where the application can easily require or leverage more threads than what is commonly available from a single computing resource. threading mechanisms include OpenMP, Cilk, and Pthread based applications, among others.
2) *Large Memory Utilization* - Where the amount of main memory needed for a given large computation can be more than what a single system can support. This includes large genomic assembly applications, or in memory database systems.

The first major use-case is large shared memory SMP systems is to support thread-level parallelism to scale beyond a single node. Often, thread level parallelism is either already available or easily obtainable with relatively minor changes to the codebase, especially when compared to the changes necessary for distributed memory solutions. As such, many threaded and OpenMP applications are easily created. In fact, application threading has become commonplace within many software systems, especially as multi-core systems have become commonplace in the past decade. However, this only allows small scaling within a single physical machine. With SMP, threads can theoretically scale across many distributed CPUs (hundreds or thousands depending on the deployment) with run-time environment changes. This effectively allows for OpenMP, Pthreads, and other types of multi-threaded applications to obtain mid-level HPC performance at relatively little cost, both in terms of physical resources and software development efforts.

The second and potentially more important use case for such an infrastructure is to support large memory applications. In many cases, fitting data directly into main memory is either impossible or infeasible on a single system, and re-writing applications from scratch to take advantage of distributed memory tools like MPI may be a far greater undertaking than a project is able to tackle. This could be due to development costs, licensing issues, or even the relatively little gains that would be achieved by traditional HPC solutions. Furthermore one could treat main memory as a tertiary storage system, in order to speed up existing database access, potentially by many orders of magnitude [6]. As such, using large SMP machines can simply these use cases by allowing for a global memory address space that spans across all nodes in the deployment without any user action or input. This makes large database applications, complicated in-memory computations, or large scale sparse memory search applications all well suited for deployment in such systems.

Cache Coherency [7] is a fundamental component of SMP systems. With any SMP, each processor or core maintains a cache of the local main memory. This is done primarily due to the fact that L1 and L2 cache speeds can be an order of magnitude faster in access times compared to main memory. However, with an SMP system it is possible to have many copies of a single memory, resulting in a problematic disparity between caches. Thus, each SMP must ensure that changes are propagated throughout the entire system as quickly as possible. There are a few coherency mechanisms that are commonly used: Directory based [8], Snooping [9], and Snarfing [10]. While all models work, they usually present various tradeoffs between latency, bandwidth utilization, and scalability [11]. While this is an active and ongoing area of research, the underlying mechanisms of Cache Coherence are beyond the scope of this paper.

Within today's SMP options, there are two shared memory systems that have entered the mainstream market: the ScaleMP vSMP platform and the SGI Ultraviolet (UV) systems.

A. ScaleMP vSMP

ScaleMP provides a novel solution for creating virtualized high performance SMP systems. The vSMP platform aggregates many commodity x86 systems, such as a classical cluster, into a single virtualized system [12], [13]. From the user and administrative perspectives, only a single, large system exists in which a patched Linux OS is deployed, usually RedHat Linux. With enough underlying hardware, this system can incorporate 128 nodes, scaling to 1024 processors (32786 cores) and 2PB of shared memory, all under just one OS.

ScaleMP's vSMP foundation software provides the backbone for producing a large shared memory system. By using a master node and multiple slave nodes, it can build a single virtual SMP machine across many distributed nodes, providing a single OS and accompanying environment. With the OS spanning multiple discrete compute resources, many of the inherent challenges of distributed systems faced by users today are greatly simplified or removed entirely. Problems such as data locality, memory management, CPU organization, etc can be left up to the ScaleMP vSMP, the OS and kernel scheduling, and vSMP support tools. This allows the programmer to easily work on a single many-core system by leveraging various threading models and focus on the computational challenge at hand, instead of spending time on the details of parallelization common with MPP systems.

ScaleMP creates this unified vSMP system by leveraging a high performance, low latency interconnect; InfiniBand. This network allows for Remote Direct Memory Access (RDMA) to be handled at the virtualization layer, and an OS via the master node is presented with a large shared memory system, even though most of the cores and memory are actually available from alternate slave nodes within the underlying cluster. Furthermore, vSMP leverages this network to ensure cache coherency between individual nodes through their proprietary coherency algorithms. This includes block-level concurrency as well as advanced caching techniques which minimize InfiniBand latency and redundant memory operations. Furthermore, the system unifies all PCI and I/O devices into the single OS, providing a large I/O device pool. This is especially useful when looking to leverage multiple accelerators within a single system, such as Intel Xeon PHIs or Nvidia Tesla GPUs, which are now supported on vSMP. For example, if you have a 16 node cluster each with 2 GPUs, you can effectively harness 32 GPUs directly from a single application deployed on the SMP OS.

Within ScaleMP vSMP, there are two fundamentally different modes that the system can be configured, system expansion mode or memory expansion mode. This configuration choice can have a drastic impact on the application being used, and must be made carefully. While it is easy to re-provision the system in either mode, such an operation requires several minutes of downtime as all master and slave nodes need to be rebooted and respective boot images distributed.

System Expansion mode in Figure 1 is specifically calibrated for compute intensive applications, whereby all the available cores are desired. Here, all the CPU and memory resources are consolidated into a single virtual system, whereby the OS can be provisioned on top. From the perspective of the OS, up to 32 thousand cores and 2PB of memory could be available, depending on the configuration.

Memory Expansion mode illustrated in Figure 2 is designed for use with applications requiring only a large amount of main memory. In this mode, only the master node's CPU and I/O devices are made available to the OS, yet all the slave's system memory is available. From the OS perspective, it looks like a normal commodity server with up to 2PB of main memory attached. While large memory applications can be run in system expansion mode, the memory expansion mode offers more efficient and simplified memory utilization through RDMA, leading to better performance and availability.

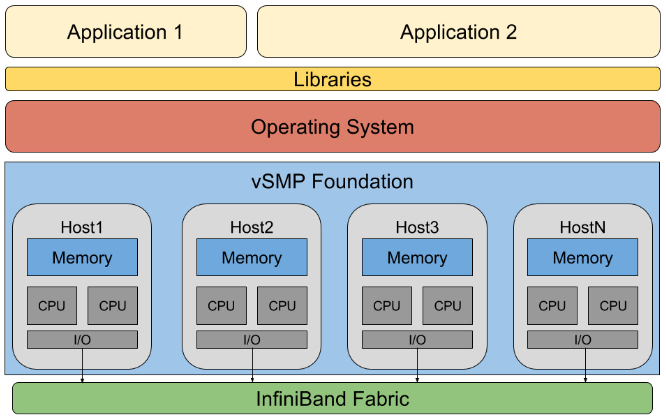

Fig. 1. vSMP System Expansion Mode

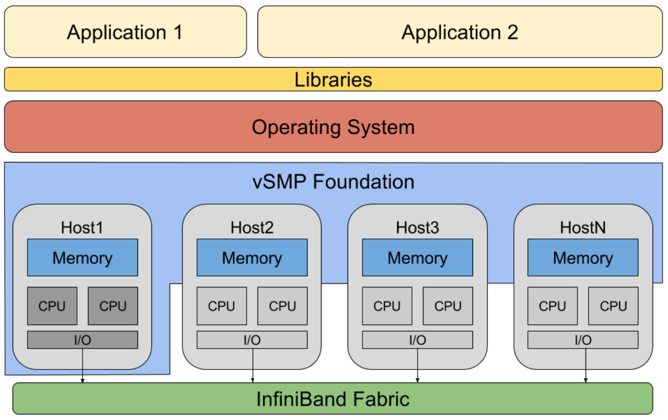

Fig. 2. vSMP Memory Expansion Mode

B. *SGI UV Altix*

SGI Corporation has been developing memory intensive systems for over 20 years. Their UV advanced symmetric multiprocessing (SMP) systems utilize NUMAlink interconnect technology to deliver scale-up performance within a single system. The SGI UV architecture can date back to the original Stanford DASH supercomputing design from the late 1980s [14]. These systems are frequently found in the high performance compute (HPC) environment for their ability to handle jobs that are too large for standard compute nodes, with comparable levels of performance even to distributed memory architectures [15].

SGI's UV architecture creates a global shared memory system using their own SGI NUMAlink interconnect [16]. From a physical standpoint, these systems are configured in a blade configuration with the specialized NUMAlink, a custom ASIC developed by SGI based on Intel Quick Path Interconnect (QPI). NUMAlink itself is made up of 4 components. A Global Register Unit expands cache coherency between blades, an Active Memory Unit that supports atomic memory operations, the Intel QPI processor interface that lets the ASIC appear as a memory controller, and the actual interconnect itself for inter-node communication. From here, the NUMAlink protocol runs atop the hardware and provides the cache coherent global shared memory system to a single OS. This protocol also allows for an MPI offload engine which offloads MPI operations to the NUMAlink ASIC instead of the CPU to speed up MPI performance within the system.

While the SGI system used in this manuscript refers to the UV1000 deployed in 2011 by the University of Arizona, SGI has updated their UV offerings. The latest model from SGI is the UV3000 which scales from 4 to 256 CPU sockets with up to 64TB of shared memory presented as a single system.

C. *Others*

There are in fact other SMP designs possible for use with shared memory big data applications, beyond the commonplace SGI UV and ScaleMP vSMP offerings. This includes new systems such as the BullX SuperNode S6000 [17], which can leverage 288 processing cores and 24TB of main memory across up to 8 modules via a specialized InfiniBand switch. Furthermore "fat nodes" or large single server architecutres are now scaling with the latest Haswell CPU architecture can scale up to dozens of cores and a few TBs of RAM. While this is not the same core count or memory capacity that's possible with SMP systems, it may be enough for many users. For applictiaons that are particularly interested in only thread-level parallelism, the latest accellerator cards may provide a similar feel to SMPs, including the Intel Many Integrated Cores (MIC) architecture known as the Xeon Phi [18]. These coprocessors allow for OpenMP codes to be automatically parallelized off the main CPU to the MICs 72 cores. However, memory availability for these systems is not comparable with or useful for large memory applications, and as such fall out of the scope of this manuscript.

III. Large Memory SMP Deployments

In this manuscript we investigate two SMP systems in detail. While these systems are very different in many aspects, they do share many similarities. While this is not meant as an apples-to-apples comparison (as such a comparison isn't realistically possible), we do hope to illustrate the nuances with each shared memory architecture in as much detail as possible.

A. *Echo at IU FutureSystems*

Initially, ScaleMP vSMP was tested using a subset of the *india* supercomputer within the NSF FutureGrid project [19]. This consisted of 16 nodes removed from normal HPC usage, creating a 128 core single system with 384GB memory ($\tilde{3}$20GB usable). Initial testing revealed that OpenMP and multi-threaded performance was relatively good, and traditional HPC workloads such as HPL and the adjoining HPCC benchmarks performed relatively well, as seen in Figure 3. Using 16 Nehalem based nodes, we found MPI efficiency to be 82.4% of peak performance. When considering this is a virtualized SMP machine, we found to be an acceptable level of overhead for many use cases. However, MPI jobs is not the primary function of the SMP machine, and more permanent deployment with a larger memory footprint was desired.

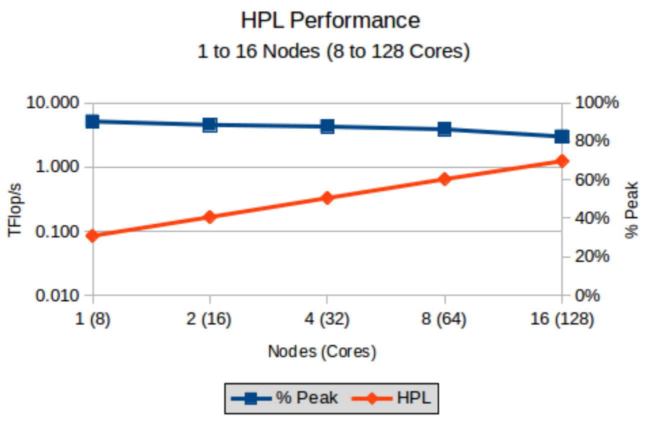

Fig. 3. HPL 128 node exploratory performance

Echo, the latest usage of ScaleMP's vSMP Foundation deployment at Indiana University (IU), provides a significant jump in computational abilities for shared memory processing in FutureGrid transitions to FutureSystems. The deployment provides 16 nodes, each with 2 Xeon Sandy-Bridge E5-2640 CPUs at 2.5Ghz and 384GB of DDR3 RAM, as described in Table III. This creates a system total of 192 cores (without Hyperthreading) and 5.6TB of usable main memory, with 470GB of main memory reserved for vSMP cache. Each node is connected using Mellanox Connect-X3 InfiniBand VPI adapters with QDR 40Gbs links across a Voltaire QDR InfiniBand switch. RedHat Enterprise Linux 6.5 is deployed as the single OS, and is managed in the *echo* queue as part of the larger Moab/Torque HPC queuing system at IU. The Echo system is loaded with vSMP foundation 6.0.135.46, which was found to be stable across all 16 nodes for a period over 1 year.

B. *UV at Arizona*

The UV system at University of Arizona (UofA) represents a common offering from SGI, and serves the many big data and large memory computing tasks at UoA. This UV1000, which is also described in Table III has eight-core Intel Xeon e7-8837 Westmere processors and a mix of 2GB and 8GB

	ScaleMP	**SGI UV1000**
Machine Name	Echo	UV
Nodes	16	16
NUMA Nodes	32	32
Cores	192	256
System RAM	5.6 TB (6.0 TB)	1.28 TB
CPU Type	Intel x86_64	Intel x86_64
CPU Model	Xeon E5-2640	Xeon E7-8837
CPU Family	SandyBridge	Westmere
CPU Base Frequency	2.5 Ghz	2.66 Ghz
CPU Max Frequency	3.0 Ghz	2.8 Ghz
CPU Cores	6	8
Cache Size	15 MB	24 MB
Interconnect	Mellanox QDR InfiniBand	SGI NUMAlink 5
IC Bandwidth	40 Gbps	15 Gbps
OS Type	RedHat EL6 Linux	RedHat EL6 Linux
OS Kernel	2.6.32-220.23.1.vSMP.el6.x86_64	2.6.32-358.6.2.hz100.el6.x86_64
SMP Version	vSMP 6.0.135.46	SGI Accelerate 1.6 SGI Foundation 2.8
Compiler	Intel 2013.sp1.2.144	Intel 2013.5.192

TABLE I
HARDWARE CONFIGURATION

DIMMs. There are 58 dual socket compute nodes for a total of 928 cores, and 2.77TB of usable memory. Each node is connected with the NUMAlink 5 interconnect system, a proprietary fiber optic interconnect provided by SGI with a line bandwidth of over 15 Gbs (two 7.5Gbs unidirectional links). The UV system at UofA is rated at 18.90 TFLOPs and is installed with Red Hat 6.0, and is controlled by the Altair PBS Pro scheduling system under the *smp_standard* queue. For the purposes of this study, the evaluation jobs were run as part of the normal operation of the system. There was not exclusive access provided over the regular workload; however, PBS Pro keeps one job from interfering with the performance of another.

IV. Experimental Setup

A. SPEC OpenMP

The Standard Performance Evaluation Corporation (SPEC) is a major standard for evaluation of benchmarking systems. SPEC has several different testing components that can be utilized to benchmark a system. For our benchmarking comparison we will use the SPEC OMP2012 [20] because it appears to represent a vast array of new and emerging parallel applications while simultaneously providing a comparison to other SPEC benchmarks. SPEC OMP continues the SPEC tradition of giving HPC users the most objective and representative benchmark suite for measuring the performance of SMP (shared memory multi-processor) systems.

The SPEC OMP2012 benchmark suite consists of a number of real world applications, all implemented using OpenMP. These applications include applications in molecular dynamics, computational fluid dynamics, sequence alignment, LU factorization, image processing, and various other simulators and solvers. These applications, originally derived from the SPEC OMP2001 suite, have multiple workload levels to characterize the performance of medium and large sized systems. Together, these applications are able to place a heavy demand on both systems and memory, key components of any SMP machine. The SPEC OMP2012 suite uses a reference system, a Sun FireX4140 with two AMD Opteron 2384 processors (quad core) to give a base score of 1 for each run. As such, all scores listed are based on the performance of this system, not a per-core scaling metric.

With deploying OMP2012 on the Echo ScaleMP system at FutureSystems, we used the vSMP System Expansion mode. This allows for access to all 192 cores, to give the maximum OpenMP performance. For the SGI UV at UoA, we leveraged a subsystem with 240 available cores, however all benchmarks were capped at 192 cores (24 NUMA nodes). For the vSMP, the KMP_AFFINITY directive was used to pin each thread on its own core. For the SGI UV, the dplace SGI tool was used to handle OpenMP thread placement. Each OMP2012 benchmark was run a total of 3 times with the full training sets, as per the run-time guidelines mandated for all reportable SPEC results, available online [21]. Again, while this is not an apples-to-apples comparison, we hope to make the results as relatable as possible. :

B. Trinity De-novo Assembly

A transcriptome is the set of all RNA molecules that are transcribed in a cell or population of cells. Transcriptomes can help to identify genes, alternative splicing in genes, and differential expression of genes in distinct cell populations. High volume RNA Sequencing (RNA-Seq) produces many millions of short RNA sequence reads. The de novo assembly of these reads into a transcriptome is a computationally intensive task. Typically these huge data sets are down-sampled to produce a smaller data set that can be assembled with commonly available computing systems. In this experiment, we assembled a full RNASeq data set of 370 million reads using the Trinity assembly package. This effort represents a real world big data problem in Bioinformatics around the world today.

The Trinity software, developed at the Broad Institute and the Hebrew University of Jerusalem, represents a novel method for the efficient and robust de novo reconstruction of transcriptomes from RNA-seq data. Trinity combines three independent software modules: Inchworm, Chrysalis, and Butterfly, applied sequentially to process large volumes of RNA-seq reads [22], [23].

When using Trinity, we created two different Trinity jobs to test the system. Both data sets use the *Crangon crangon* (Shrimp) transcriptome, gathered from an Illumina RNAseq dataset. the reads are around 100bp in length, and was trimmed to approximately a 50x coverage using the khmer normalization tool. In Job 1, only the R1 reads are used, whereas in Job 2, assembly was performed using both the R1 and R2 reads. Because Trinity is mainly a memory intensive application, especially given our big data set, the vSMP system was set up this time in Memory Expansion mode. This mode allows for a performance boost over system expansion due to the vSMP RDMA and caching techniques.

V. Results

The goal of this manuscript is to effectively compare and contrast the ScaleMP vSMP deployment at IU, and the SGI UV1000 deployment at UofA, specifically for use with OpenMP threaded and big-data applications. The first set of results represent the SPEC OpenMP benchmark suite, and the second set of results relate to the Trinity runs, both described in Section 4.

A. OpenMP

In Figure 4, we see the SPEC OpenMP 2012 benchmark suite results immediately paint a very complex picture when it comes to the performance. In part, this is due to the 14 different applications incorporated in the OMP2012 suite, each with vastly different attributes. The *y* axis represents the SPEC score retrieved for each individual benchmark, which is a ratio of the benchmark run-time in relation to the SPEC reference system. For clarity, a spec score of 5 illustrates how that particular application ran 5 times faster then the reference system.

From Figure 4, we can see with the variance in benchmark SPEC scores that some benchmarks inherently scale better than others. For instance, the 352.nab, 362.fma3d, 370.mgrid, and 371.applu331 exhibit signs of weak scaling in both the vSMP and UV systems. This is largely due to the inherent implementation of each benchmarks' algorithm when running in such SMP environments. However, some applications such as 350.md, 358.botsalgn, 372.smithwa, and 376.kdtree, scale well on both systems, reaching up to a 53.6 SPEC score for the UV. This shows that the SMP machines are able to provide large speedups with OpenMP applications, however not all OpenMP applications will automatically scale well. This is the case for many parallel applications independent of implementation, as Ahmdahl's Law [24] impacts parallel applications differently, not to mention the way in which each algorithm is implemented can directly impact scalability. This is especially true with OpenMP, which leaves many parallelization details to the compiler and run-time environment, in contrast to the fine-grained control of the developer as with MPI applications.

As a major goal of this manuscript is to compare these differing SMP systems, we must dive further into the SPEC OMP2012 results. The total SPEC base score for the ScaleMP vSMP and SGI UV1000 is 10.4 and 7.01, respectively for 192 core runs. This score is a geometric mean of the 14 normalized SPEC ratios, and give a quick summary of each machine's overall performance. While this may be a quick way to judge performance and assume the VSMP is faster, there is more to the story, as SMP machines incorporate complex non-deterministic architectures. As such, we look to group the benchmarks based on the two system's performance to better understand each SMP system's behavior.

Of the 14 total benchmarks within the SPEC OMP2012 suite, 9 benchmarks were relatively comparable between the two SMP systems, with the vSMP system performing better in 5 of the 9 benchmarks. If we consider just these 9 benchmarks and calculate the geometric mean of the SPEC scores (the same function as the total SPEC base score), the vSMP gets a score of 7.65 compared to the UV's score of 7.44. As both benchmarks were run with 192 cores, this indicates these benchmarks are largely CPU bound and relative to core count and performance.

There were some benchmarks that showed drastic difference between the two systems. The vSMP system significantly outperformed the UV in 4 benchmarks in particular, 351.bwaves, 360.ilbdc, 363.swim, and 370.mgrid331, where the vSMP was at least almost double the speed of the UV. The worst case was 351.bwaves, where the vSMP was 12.8 times faster than the UV. In contrast, the UV1000 at UofA significantly outperformed the vSMP in 1 benchmark, the 372.smithwa, where the UV scored a 53.6 score to the vSMP's 19.3. Here, the UV is 2.7 times faster for the Smith-Waterman benchmark, an algorithm commonly implemented as BLAST alignment [25]. As this benchmark has a particularly small memory footprint, it seems that the UV is able to take advantage of its additional cores per socket and larger cache to provide significantly better performance compared to the vSMP at IU.

While the initial results point to the ScaleMP vSMP deployment outperforming the SGI UV at 192 cores, this is not surprising given the details of each deployment. When considering each system's CPU architectures, described in Table 1, we see the vSMP consists of Sandy-Bridge based E5-2640 CPUs in dual-socket 6 core configuration, with a total of 12 cores per node. This compares to the SGI UV1000's system, where each blade uses older Westmere based E7-8837 CPUs, also in a dual-socket configuration, except each socket has 8 cores for a total of 16 cores per UV blade. In order to compare each CPU's performance, we can leverage the SPEC CPU2006 benchmark [26], which provides CPU-intensive SPEC scores for single node systems. From the SPEC CPU2006 results [27], we find two benchmarks results

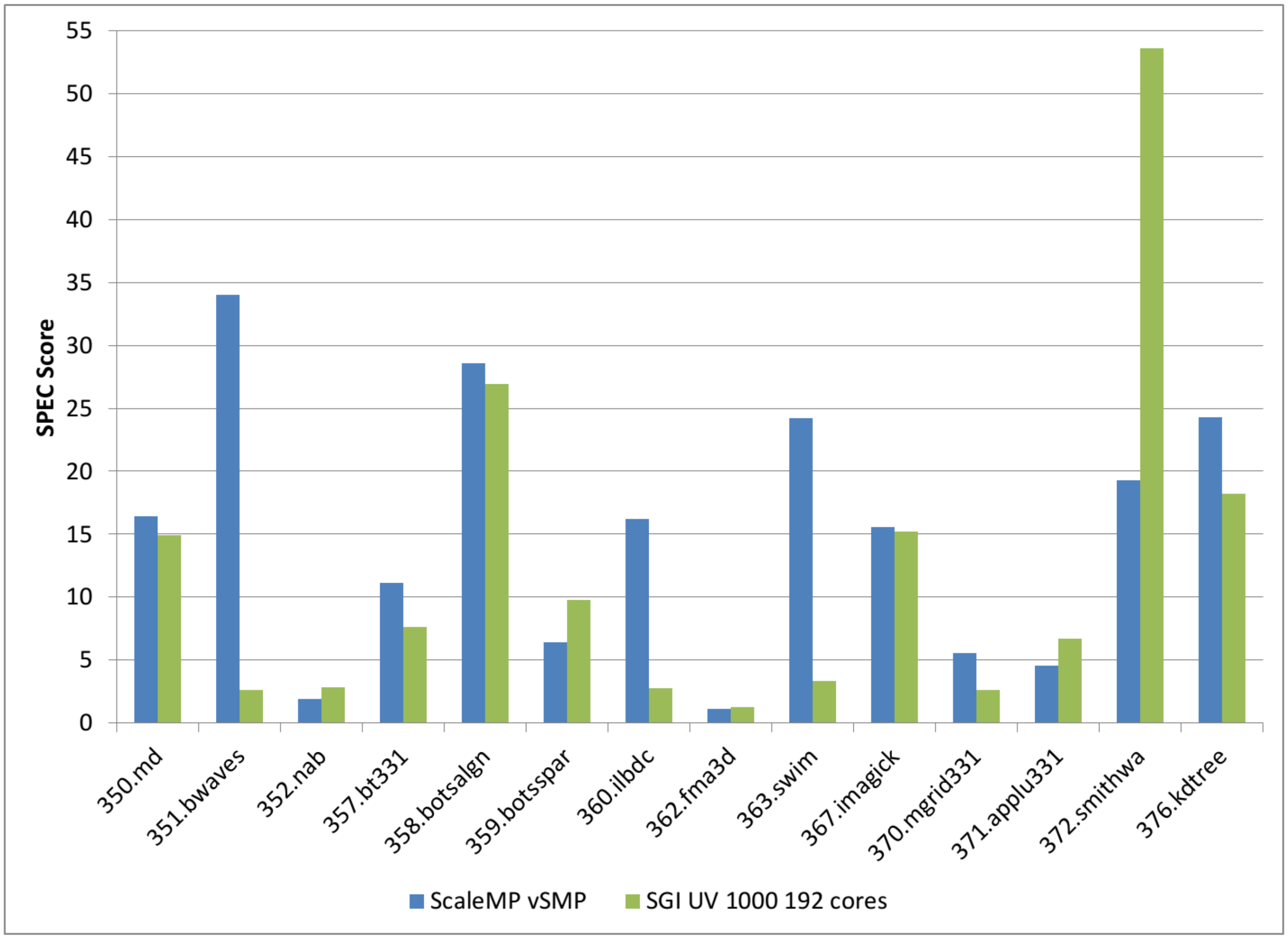

Fig. 4. SPEC OpenMP 2012 Benchmark suite

from Supermicro Inc, models 6017TR-TQF and 5086B-TRF, which have identical CPUs and NUMA sockets as our SMP systems. The Sandy-Bridge E5-2640 reports a score of 46.0, whereas the Westmere E7-8837 manages a score of 39.3. From this, we can infer that the Sandy-Bridge CPUs in the ScaleMP are roughly 1.17 times, or 17% faster than the UV's Westmere CPUs.

If we were to assume this 17% performance improvement inherent in the CPU architecture of the vSMP carries linearly to the SMP machines, one can expect the SPEC OMP2012 scores to also exhibit the same difference. However, with total SPEC base scores of 10.4 and 7.01 for the vSMP and SGI UV, respectively, we can infer the vSMP is about 1.5 times faster than the UV overall. While the Sandy-Bridge CPU architecture improvement in the VSMP can explain some of this performance, there is more to understand.

If we take the 4 benchmarks where the vSMP outperformed the UV, 351.bwaves, 360,ilbdc, 363.swim, and 370.mgrid331, and calculate the geometric mean the spec scores for each system, we find the vSMP outplaces the SGI by 5.8 times. This shows there is a very large difference between the two systems that can occur. Looking at these 4 benchmarks in more detail from the SPEC OMP2012 publication [20], we see each application has a high memory footprint, indicating there is a substantial amount of memory transfers occurring between each SMP system's nodes. These RDMA memory accesses occur across the QDR InfiniBand network for the vSMP at a bandwidth of up to 40Gbs, whereas the SGI UV uses the NUMAlink5 interconnect with only a 15Gbs peak bandwidth through 2 unidirectional 7.5Gbs links. Looking at the unidirectional link bandwidth, we see the vSMP has 5.33 times more bandwith than the SGI UV. This may be further compounded by the fact that 16 cores share the same bandwidth NUMAlink route on the SGI UV, compared to only 12 cores sharing a QDR InfiniBand link, inferring there may also be greater interconnect contention on the SGI UV. This inter-node bandwidth discrepancy, coupled with the architecture differences, likely explains why the vSMP significantly outperforms the UV in the 4 memory-intensive OpenMP benchmarks.

B. Trinity

Figure 5 shows the fundamental aspect of running any application: wallclock time. In Figure 5 we can see the runtime of two jobs with both the ScaleMP vSMP in memory mode and Arizona's SGI UV 1000. Here, we notice that the vSMP outperforms the UV1000 by 15.6% for Job 1 and 9.3% for Job 2.

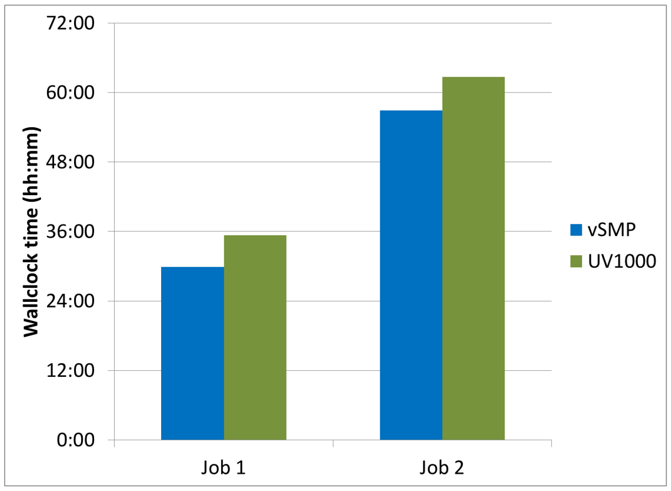

Fig. 5. Trinity Assembly Wallclock Time

Interestingly, Figure 6, which shows total CPU calculation time, is not at all representative of the outcome of the actual wallclock running time. This is due to the fundamental architectural differences between the two systems. With vSMP in memory mode, it considers all non-local (RDMA) memory accesses to be system I/O and therefore not counted towards the first processor's CPU time. With the SGI UV, however, we see that the opposite is true, as remote memory accesses are processed by those CPUs which control the respective memory space, leading to a higher CPU time. The difference between vSMP and the SGI in Job 2 shows that much of Job 2's workload is actually spent accessing remote memory, and not in direct algorithmic calculations. The RDMA offload engines within the Mellanox Connect-X3 InfiniBand adapters are likely responsible for this, as memory buffering happens without direct CPU involvement and only a CPU interrupt.

The key aspect and utility of these large memory SMP machines is just that - their ability to access large amounts of memory. As such, it is imperative to consider whole memory usage with the two workloads deployed in the Trinity application. In Figure 7, we see the resident and virtual addressing that each job uses. First and most fundamentally, we see that Job 2 accesses over 1 Terabyte of main memory, which makes the existence of such large memory systems of pivotal importance. Looking deeper, we can also see some architectural differences in the way in which each system handles remote memory. With the vSMP system in memory expansion mode, the OS shows most of the memory is resident to the local CPU. However, with the many-core availability

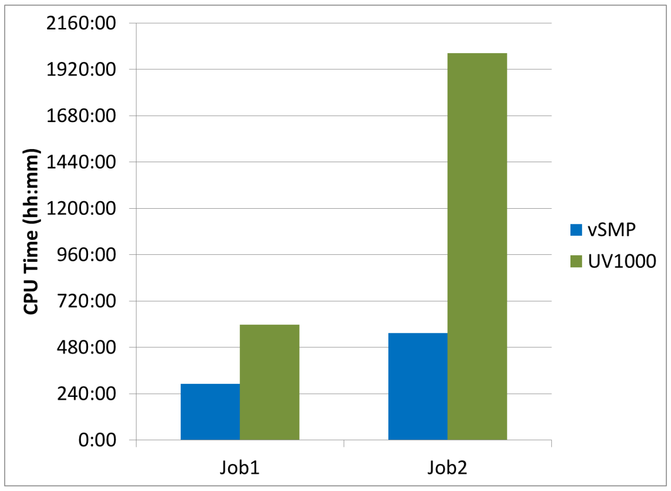

Fig. 6. Trinity Assembly CPU Time

in the SGI UV architecture, resident memory is listed only based on the controller board's memory usage, which is 4.3% and 12.7% the total virtual memory space for Job 1 and Job 2, respectively. While this may be a matter of OS accounting semantics, it does appear that there is a significant overhead in virtual memory utilization with the vSMP memory expansion mode. This overhead decreases as overall resident memory usage increases, with Job 1 requiring 23.6% virtual memory space and Job 2 requiring an extra 8.1% of virtual memory space when compared to the UV1000. While this is unlikely to be an issue for most users, it is potentially worth budgeting in this overhead when considering overall memory usage in a ScaleMP system.

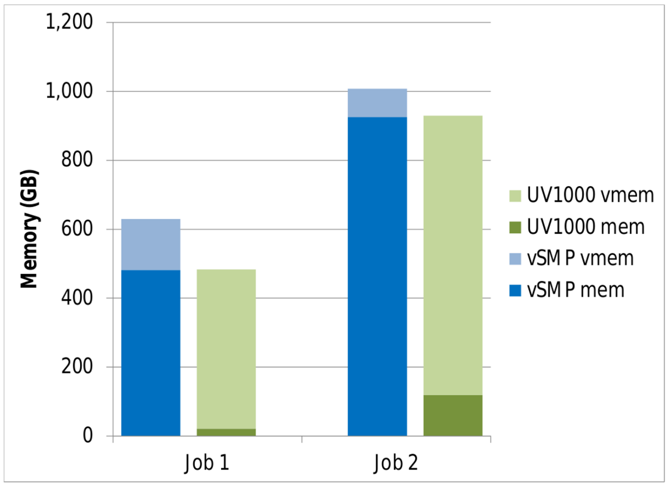

Fig. 7. Trinity Assembly Memory Utilization

VI. Discussion

From the SPEC OMP2012 benchmark suite focusing on OpenMP multi-threaded performance and the Trinity de novo assembly application, we've found a number of performance implications for both vSMP and UV systems. From OMP2012 benchmarks as a whole, we've found that the vSMP generally performs better at 192 cores than the SGI UV. Much of this performance improvement of the vSMP over the SGI UV is expected as the vSMP utilizes a newer CPU architecture, and accounts for vSMP running faster on the majority of benchmarks. However, there some benchmarks where the vSMP significantly outperforms the UV, sometimes by margins beyond 5 times. This is most likely due to interconnect bandwidth contention that exists in the UV across the NUMAlink5 architecture when compared to the faster QDR InfiniBand interconnect found with ScaleMP's vSMP. This difference plays a significant role in performance for applications that are bound by memory bandwidth, but not for CPU bound tasks where there is little inter-node communication and remote memory access. If the per core memory usage is especially small and the application is almost purely CPU bound, the SGI UV performs much better than the vSMP, as seen with the Smith-Waterman benchmark. It is our hypothesis that this is due to the cache size differences between the two CPU architectures that favors the Westmere based SGI UV1000. It is also worth noting that several OMP2012 benchmarks did not scale much past 2x performance speedup compared to the reference system, indicating very weak scaling of the benchmarks themselves. This goes to show that not all applications will effectively parallelize with OpenMP, and that scaling is still largely dependent on the algorithm itself. Furthermore, the overhead of remote memory accesses increases as the systems get larger [28]. It is also worth stressing that we are only characterizing two deployed systems, which may not accurately represent other current or future ScaleMP vSMP or SGI UV deployments, as there exists many differences between each deployment.

From Trinity, we first learned that both the ScaleMP vSMP and SGI UV1000 are capable of handling large in-memory big data applications. Specifically, the Trinity Job 2 utilized over 1TB of main memory for the transcriptome assembly, a significant feat that is largely unmanageable by most commodity systems today. Furthermore, the traditional code of Trinity was not change or altered in any way, showing how off-the-shelf software that hasn't been rewritten for use in a distributed memory cluster can still scale to a necessarily large size. When comparing the vSMP to the SGI UV, we again find the vSMP slightly faster than the UV, with a 9.3% reduction in wallclock time for the larger Trinity job. As we expect the coverage of the transcriptomes to increase dramatically in time, this paves the way for even larger Trinity jobs in the future.

A. Usability

Both SMP systems evaluated in this manuscript provide mechanisms to easily scale computation to hundreds of CPUs and terabytes of memory in a manner that is far simpler than traditional distributed system deployments. However, both systems still require some effort to properly tune applications to best utilize the available resources. Often unmodified applications deployed on vSMP with no tuning can have drastically decreased performance than optimal. This is largely due to non-uniform memory access (NUMA) issues inherent with a shared memory system, as well as the inability of the OS scheduler to adapt. As such, the ScaleMP manual was developed for FutureSystems to help new users leverage the power of the *Echo* system [29]. Applications are classified into MPI, threaded, OpenMP, throughput, and serial types, each with an example use case to show how the application can be quickly tuned to gain optimal performance. This piece of documentation can be critical for effectively using the shared memory resource because most of the shared memory use cases come from users who have little detailed knowledge of the parallelism in the applications. For instance, a user may want to use a commercial toolkit, but their problem set requires a large memory space and they have no ability to modify or adapt the application for a more classical distributed memory cluster or supercomputer. In such situations, the SMP resource fits well, but some application tuning is still needed. Luckily, this tuning effort for either ScaleMP vSMP or an SGI UV machine is likely far less than rewriting the given workload to utilize a classic distributed memory HPC environment.

While not a typical use case for SMP machines, both the ScaleMP vSMP and SGI UV can also support traditional HPC workloads, including MPI distributed memory applications. As referenced in Figure 3, there is a slight but notable overhead compared to a bare metal distributed memory cluster for running MPI applications such as Linpack on vSMP. The SGI UV1000 offers a specialized MPI offload engine as part of the SGI foundation software. This ability for these SMP systems to support any workload, threaded, large memory, or even MPI based, is a considerable advantage for supercomputing centers, as workloads are often very diverse.

As the IU FutureSystems goal is to provide an experimental test-bed for Grids, Clouds, and HPC, we are at an ever-evolving intersection of application prototyping and development across a diverse set of scientific fields. As such, the Echo vSMP deployment found highly variable usage requests throughout the project. At times, multi-user scheduling necessary on *Echo* to enable multiple users to simultaneously develop and deploy their codes on Echo. At other times, Echo's usage was relatively low with many idle cycles. This is due to the dynamic provisioning aspects of ScaleMP's vSMP, which were always desired but never deployed.

An advantage of ScaleMP vSMP is that the underlying hardware is commodity x86 based, connected with an InfiniBand interconnect and could, with proper provisioning tools, be redeployed as a commodity distributed memory cluster in a matter of minutes, allowing for maximum utilization of the hardware independent of the workload type. Ideally, one could have a large shared memory machine available on demand when needed, but also allow the hardware to serve more traditional HPC workloads when large memory requirements didn't exist. While this is not possible using an SGI UV architecture, the SGI is still known for supporting MPI workloads effectively [30].

VII. Conclusion

While distributed memory clusters and supercomputers are a common solution for big data science, there are times when large shared memory SMP machines are needed. This includes instances when rewriting the application or workload is either impractical, infeasible, or outside the control of the scientists. In this manuscript, we have evaluated SMP machines that have the ability to support big data workloads; the ScaleMP vSMP deployment at Indiana University, and the SGI UV1000 deployment at University of Arizona. While these machines are different in terms of both design and implementation, they do represent relatively similar possibilities and size.

In this study, both SMP machines perform very well at both OpenMP threaded tasks and with our big data bioinformatics de novo assembly application, Trinity. We found that overall the ScaleMP vSMP system deployed at Indiana University often performed better at OpenMP applications. Many CPU-bound applications performed slightly faster on the vSMP deployment, and some memory-bound OpenMP codes performed far better in some special cases. We also found improved performance with the ScaleMP vSMP running Trinity de novo assembly tasks, largely relating to the CPU architecture improvements realized by the hardware in the ScaleMP deployment. In summary, both SMP deployments are well suited to provide exceptional computing environments for supporting big data workloads which don't fit to classic distributed memory architectures. Due to the increased computing demand seen from the explosion of big data science and the relative ease of use with both the ScaleMP vSMP and SGI UV, we expect the use of both systems to only increase in the future.

Acknowledgment

This document was developed with support from the National Science Foundation (NSF) under Grant No. 0910812 to Indiana University (IU) for "FutureGrid: An Experimental, High- Performance Grid Test-bed." Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF. Andrew J. Younge also acknowledges support from the Persistent Systems Fellowship of the School of Informatics and Computing at IU. The authors would also like to acknowledge the FutureSystems team at IU for their help with system setup, and Susan J. Miller for her help in formulating the Trinity de novo assembly data sets.

References

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE '08*, Nov 2008, pp. 1–10.

[2] J. Dongarra, H. Meuer, and E. Strohmaier, "Top 500 supercomputers," website, November 2008.

[3] C. Schimmel, *UNIX systems for modern architectures: symmetric multiprocessing and caching for kernel programmers*. Addison-Wesley Longman Publishing Co., Inc., 1994.

[4] A. J. Hey, S. Tansley, K. M. Tolle *et al.*, *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research, 2009, vol. 1.

[5] J. Shun and G. E. Blelloch, "Ligra: A lightweight graph processing framework for shared memory," in *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '13. New York, NY, USA: ACM, 2013, pp. 135–146.

[6] K. Goda and M. Kitsuregawa, "The history of storage systems," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, May 2012.

[7] D. J. Sorin, M. D. Hill, and D. A. Wood, "A primer on memory consistency and cache coherence," *Synthesis Lectures on Computer Architecture*, vol. 6, no. 3, pp. 1–212, 2011.

[8] D. Chaiken, C. Fields, K. Kurihara, and A. Agarwal, "Directory-based cache coherence in large-scale multiprocessors," *Computer*, vol. 23, no. 6, pp. 49–58, 1990.

[9] C. Ravishanicar and J. R. Goodman, "Cache implementation for multiple microprocessors," 1983.

[10] P. L. Borrill, "System and method for a snooping and snarfing cache in a multiprocessor computer system," Dec. 24 1996, uS Patent 5,588,131.

[11] S. V. Adve and K. Gharachorloo, "Shared memory consistency models: A tutorial," *computer*, vol. 29, no. 12, pp. 66–76, 1996.

[12] A. Snell, "Outward and upward: Recreating scalability with vSMP foundation," Tech. Rep., Feb 2013.

[13] N. Berr, D. Schmidl, J. H. Göbbert, S. Lankes, D. an Mey, T. Bemmerl, and C. H. Bischof, "Trajectory-search on scalemp's vsmp architecture." in *PARCO*, 2011, pp. 227–234.

[14] D. Lenoski, J. Laudon, K. Gharachorloo, W.-D. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam, "The stanford dash multiprocessor," *Computer*, vol. 25, no. 3, pp. 63–79, March 1992.

[15] J. Dunigan, T.H., J. Vetter, and P. Worley, "Performance evaluation of the SGI Altix 3700," in *Parallel Processing, 2005. ICPP 2005. International Conference on*, June 2005, pp. 231–240.

[16] SGI, "Performance and productivity breakthroughs with very large coherent shared memory: The SGI UV architecture," Silicon Graphics International Corp, Tech. Rep., Jan 2012.

[17] S. Bull, "BullX Supernode Specification Sheet," Webpage, 2014.

[18] J. Jeffers and J. Reinders, *Intel Xeon Phi coprocessor high-performance programming*. Newnes, 2013.

[19] G. Von Laszewski, G. C. Fox, F. Wang, A. J. Younge, A. Kulshrestha, G. G. Pike, W. Smith, J. Voeckler, R. J. Figueiredo, J. Fortes *et al.*, "Design of the futuregrid experiment management framework," in *Gateway Computing Environments Workshop (GCE), 2010*. IEEE, 2010.

[20] M. S. Müller, J. Baron, W. C. Brantley, H. Feng, D. Hackenberg, R. Henschel, G. Jost, D. Molka, C. Parrott, J. Robichaux *et al.*, "SPEC OMP2012 an application benchmark suite for parallel systems using OpenMP," in *OpenMP in a Heterogeneous World*. Springer, 2012.

[21] Standard Performance Evaluation Corporation, "SPEC OMP2012 Results," 2016. [Online]. Available: https://www.spec.org/omp2012/results/

[22] M. G. Grabherr, B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng *et al.*, "Full-length transcriptome assembly from RNA-Seq data without a reference genome," *Nature biotechnology*, vol. 29, no. 7, pp. 644–652, 2011.

[23] R. Henschel, M. Lieber, L.-S. Wu, P. M. Nista, B. J. Haas, and R. D. LeDuc, "Trinity RNA-Seq assembler performance optimization," in *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the campus and beyond*. ACM, 2012, p. 45.

[24] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*. ACM, 1967, pp. 483–485.

[25] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

[26] J. L. Henning, "Spec cpu2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[27] Standard Performance Evaluation Corporation, "SPEC CINT2006 Results," 2016. [Online]. Available: https://www.spec.org/cpu2006/results/

[28] D. Schmidl, D. an Mey, and M. S. Müller, "Performance characteristics of large smp machines," in *OpenMP in the Era of Low Power Devices and Accelerators*. Springer, 2013, pp. 58–70.

[29] A. J. Younge, "ScaleMP vSMP on FutureSystems," 2013. [Online]. Available: https://portal.futuresystems.org/manual/scalemp-vsmp

[30] S. Saini, D. Talcott, D. Jespersen, J. Djomehri, H. Jin, and R. Biswas, "Scientific application-based performance comparison of SGI Altix 4700, IBM POWER5+, and SGI ICE 8200 supercomputers," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, Nov 2008, pp. 1–12.