Exploring AVS for HPDC Software Integration: Case Studies Towards Parallel Support for GIS

Gang Cheng, Chris Faigle, Geoffrey C. Fox, Wojtek Furmanski, Bin Li and Kim Mills

Northeast Parallel Architectures Center
111 College Place, Syracuse University
Syracuse, NY 13244–4100

presented by W. Furmanski at the AVS Conference *AVS'93*, Lake Buena Vista, FL, May 24–26, 1993.

March 31, 1992

Abstract

We present here several recent results and summarize ongoing and planned projects at NPAC (Northeast Parallel Architectures Center) which explore AVS as a software integration tool for High Performance Distributed Computing (HPDC). Individual projects are used first to expose different aspects of this evaluation process and we then focus on Geographic Information Systems (GIS) and illustrate our approach towards developing massively parallel processing support in this area with AVS based user front-ends.

Contents

1. Overview

NPAC is a research center in High Performance Computing (HPC), pursuing several interconnected programs such as: a) development of core HPC technologies [BCFHR93], b) integration of parallel processing into industry, and c) evaluation of emergent high end computing technologies and standards. As one of the efforts within the last thrust, we were exploring over the period of the last two years the AVS environment, both in the context of scientific visualization and also, more broadly, as an attractive paradigm for software integration in High Performance Distributed Computing (HPDC). The system was installed on all major workstation types (DEC, SUN, IBM) at NPAC in fall '91 and used initially for several scientific visualization projects in areas such as High Energy Physics, GIS, Solid State Physics and Stock Market Analysis. In the next step, it was also used to integrate massively parallel processing modules into distributed processing applications. Finally, we also recently started to explore possible models for "parallel AVS", i.e. a HPC software paradigm for concurrent integration of dataflow and data parallel programming technologies.

As a result of these experiments, we see AVS as a viable tool for developing and integrating large scale applications in HPDC and we are now including AVS support into several new planned projects at NPAC. In the application domain, one of the major focal points and the emergent expertise at NPAC is in several areas of the Earth Sciences such as Geographic Information Systems or Environmental Modeling. We are currently at the planning stage of a large scale development project for Integrated Decision Support System for Environmental Policy/Analysis. This system would be based on the massively parallel support for GIS and would integrate several functional modules such as atmospheric deposition, soil chemistry, land use management, elevation, hydrology, spatial interpolation and public policy analysis.

Several modules listed above would be provided by other research labs. NPAC's initial focus in this large endeavor is on: a) software integration tools to provide required connectivity between system modules, b) human-interface front ends such as advanced visualization and Virtual Reality, c) massively parallel support for core GIS services, to be implemented on a cluster of NPAC parallel systems including CM-5, nCUBE2 with parallel Oracle, DECmpp and iPSC.

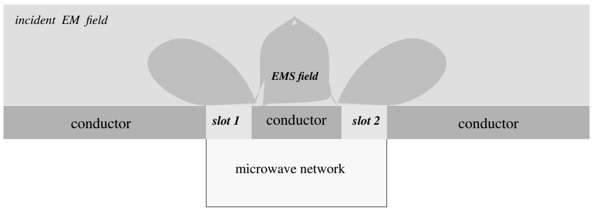


Figure 1: Profile of the electromagnetic scattering problem

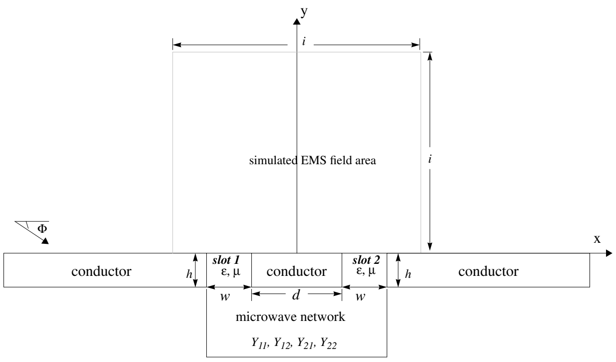


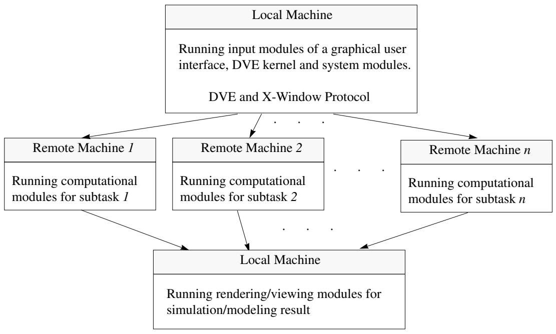
Figure 2: Physical parameters of the EM scattering problem

We see AVS as an attractive software technology in this context and we present in this paper the current status of various research projects at NPAC, exploring the individual software engineering challenges listed above. We start in Section 2 with a study of using AVS to convert a single existent large parallel code into an interactive HPDC application. We next explore in Section 3 the advanced 2D and 3D visualization capabilities of AVS applied to selected GIS filters. In Section 4, we address the issue of parallel GIS support by analyzing mappings between major categories of raster GIS functions and corresponding data parallel algorithms. Finally, we address in Section 5 the challenging issue of “parallel AVS” and we propose a dynamic model which supports concurrent integration of the AVS-type functional dataflow techniques with the HPF-type data parallel programming techniques.

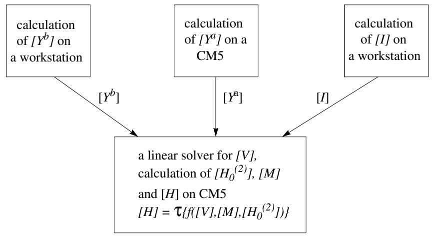
2. AVS with MIMD-Parallel Modules

We discuss first the recent results on Electromagnetic Scattering (EMS) simulation on NPAC’s CM-5 where AVS was used to integrate the CM Fortran simulation code with the distributed front-end including I/O devices, problem parameter generators and visualization desktop [CLFMH93].

EMS is a widely encountered problem is electromagnetics with important applications in industry such as microwave equipment, radar, antenna, aviation, and electromagnetic compatibility design. Fig. 1 illustrates the problem we are modelling. Incident EM field scatters on two slots on the conducting plate, interconnected to a microwave network behind the plane. The goal is to calculate the strength of the scattered field under various physical circumstances. Fig.2 summarizes the geometry and physics parameters of the problem. The presence of two slots and the microwave load requires high performance computing and complex problem geometry requires advanced visualization support.



Figuse 3: A general performance model for a remote visualization enviroment



Figuse 4: Decomposition of the simulation task

The moment method is used as the numerical model of the EMS problem, which can be represented as:

$$\{[Y^a] + [Y^b]\}\vec{V} = \vec{I} \tag{1}$$

$$[H] = L\Big\{f\Big(\vec{V}, \vec{M}, [H_o^2]\Big)\Big\} \tag{2}$$

where $[Y^a]/[Y^b]$ are equivalent admittance matrices for the free space / microwave network, $\vec{V}/\vec{I}$ is coefficient / excitation vector, $\vec{M}$ is a vector of mode functions, $[H_o^2]$ is a matrix of Hankel functions, L is a linear operator and $[H]$ is final matrix of the simulated EMS field strength.

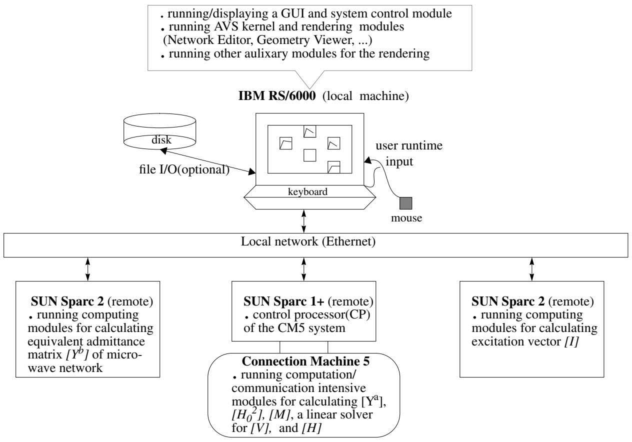


Figure 5: System configuration and distributed components

Detailed analysis of the computational task sketched above indicates that: a) computations of $[Y^a]$, $[Y^b]$, $\vec{I}$, $\vec{M}$, $[H_o^2]$ can be done independently, b) computation of $[Y^a]$, $[H_o^2]$ and the linear solver for $\vec{V}$ have significant communication requirements and are computationally intensive, and c) $[Y^b]$ is a sparse matrix and calculation of $\vec{M}$ requires little time. Calculation times for $[Y^a]$, $[Y^b]$, $\vec{I}$ are relatively balanced. Based on this analysis, we decomposed the simulation tasks as illustrated in Fig. 4.

Table 1: Timings of calculations and communications (in second)

Module name	Calculation time	Communication time with 'Input-interface-IBM'	Communication time with 'EM-all-CM5'	Communication time with 'EM-3D-IBM'
EM-1-SUN	0.1 (Sun)	0.02	0.045	
EM-2-SUN	0.6 (Sun)	0.02	0.5	
EM-3-CM5	1.8 (CM5) 1260 (Sun)	0.02	0.0	
EM-all-CM5	2.1 (CM5) 120 (SUN)			0.5
AVS Rendering Modules	3.5 (550x580 window)			

Note: The timing data is measured at a typical case: the number of pulse functions = 64; the number of mode functions = 5, and the number of grid points = 101x101.

A generic cycle for an AVS dataflow based distributed simulation is presented in Fig. 3 and the specific implementation for the EMS problem is shown in Fig. 5. Simulation parameters (see Fig. 2) are set on the local machine (in our case IBM RS/6000) to activate a processing network (in our case two Sparc stations and the CM-5) and return visual feedback to the local node.

Preliminary performance analysis is shown in Table 1. Our experiments showed that under a typical network load conditions, a complete simulation cycle for the EMS problem takes about 8 sec, which is a satisfactory response time for this application.

The CM Fortran code for the EMS scattering used in this experiment was developed earlier in another research project. Our focus in this project was on evaluating AVS as a system integration tool for HPDC applications. The results are encouraging and we intend to apply similar irregular problem decomposition and task allocation techniques to the planned GIS applications.

3. Exploring AVS for Selected GIS Functions

In another research project at NPAC, we were analyzing image processing algorithms for color separation of noisy RGB images of terrain maps and we developed an AVS based tool, integrating system's 2D and 3D visualization modules with the computational modules, developed previously within this 'Map Separates' project [Furm92].

The problem was posed by the Defense Mapping Agency (DMA) and was addressed by NPAC in collaboration with the Cartographic Group at Caltech/JPL. DMA maintains a huge collection of maps from various regions of the world, some of them stored in the form of video tapes. The agency explores now various strategies for converting this collection into an electronic database. The physical maps were often poor quality (e.g. old) and the procedure of videotaping and then reconstructing the digital raster introduced additional distortions so that the map images to be color separated are typically inflicted with substantial noise in the color space. Fig.6 illustrates a typical map image to be separated into seven base colors. The original image is in RGB format and in fact the full spectrum of colors is present there due to the noise effects.

To solve the color separation problem, we developed a class of RGB clustering algorithms, similar to conventional gray level histogramming techniques but extended to 3D color space. An RGB histogram in the unit color cube for the map in Fig 6. is shown in Fig. 7. A given RGB bin is represented by a sphere with a radius proportional to the bin context. Clean color patches would be mapped into isolated spheres, whereas the noise manifests in the form of color clusters with various, often complex shapes. Poor color saturation of the target image manifests as clusters' elongation along the R = G = B axis. The separation process must be human guided due to the fact that several complex cartographic elements such as shaded relief are to be ignored in the separated process.

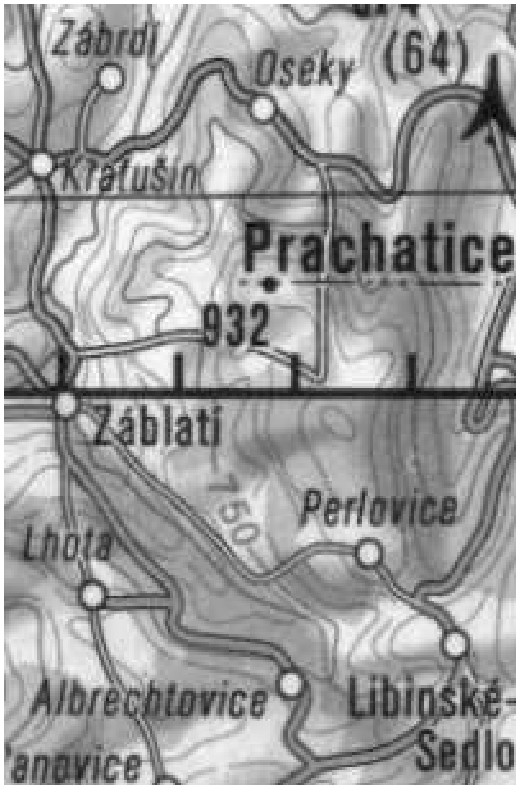


Figure 6: Terrain map image, used as input for the RGB clustering based color separation algorithm.

The RGB cluster based human guided separation procedure can be viewed as an interactive K-means clustering algorithm i.e. it consists of selecting several 'typical' small color patches on the map and assigning one of the seven base colors to each selected group of pixels. The RGB centers of all selected patches are accumulated and used to separate the volume of the color cube into a set of polyhedral regions, representing the assigned base colors. For a typical class of DMA maps, this algorithm saturated after ˜50 selections. The resulting polyhedral partition of a color cube was then used as a lookup table to assign the base color to each image pixel. The resulting separated image is shown in Fig 8.

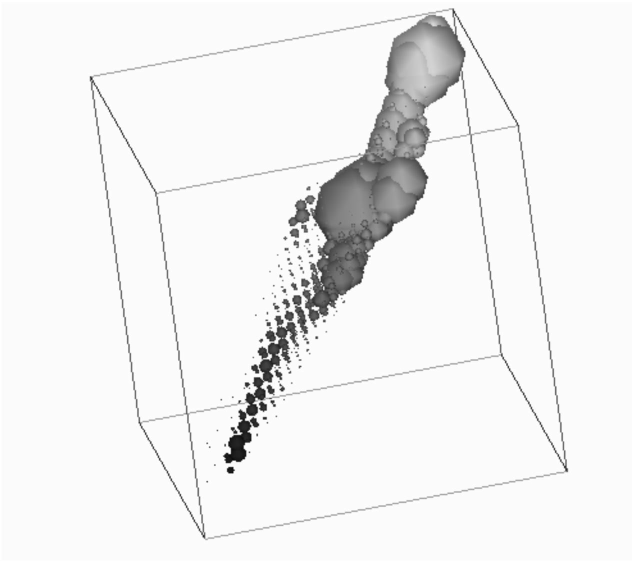


Figure 7: Color histogram of the terrain map image in Figure 6.

The AVS based tool, constructed to solve this problem exploited both the Image Viewer (for image manipulations) and the Geometry Viewer (for RGB cluster manipulations). We found the Geometry Viewer extremely useful for interactive visualization and inspection of complex 3D cluster shapes. In fact, this visual analysis turned out to be the turning point in the project as it allowed us to construct immediately the optimal clustering algorithm. The tool was implemented in fall '91 using DEC AVS 3.0 on the DEC PXG Turbo system with the 3D graphics accelerator and it was the first use of AVS for NPAC projects.

4. Parallel Processing Support for GIS

Raster GIS is a natural application area for massively parallel processing due to direct mapping between raster coverages and parallel arrays. We have performed recently a systematic analysis of computational and algorithmic complexity of all base raster GIS primitives, as classified in [Toml90], from the perspective of their parallel implementation [Li92]. Within the operation oriented classification, GIS functions are typically split into *local*, *focal*, *zonal* and *global* operation. Data parallel operations can be split into: a) common operators in sequential languages, b) elementwise operations (e.g. array arithmetic), c) mixed operations between parallel and scalar variables (e.g. reduction, distribution, insertion or extraction), d) selection (WHERE, FORALL), and e) communication functions (moves, combines, permutations and scans).

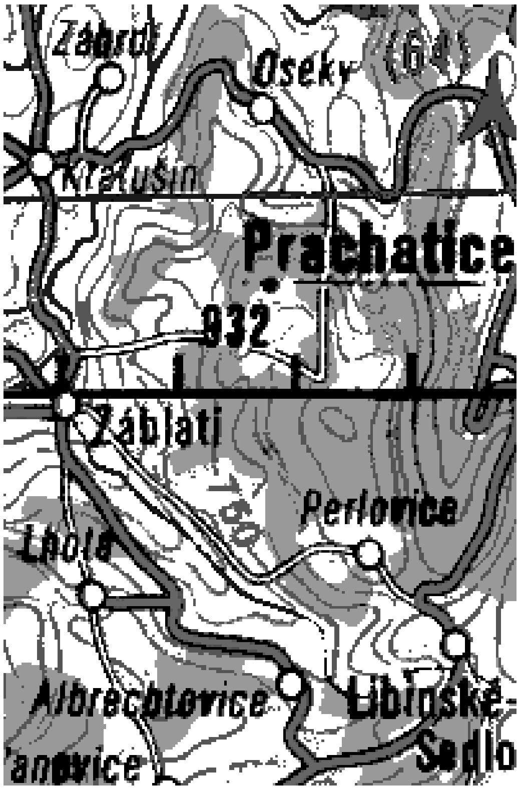


Figure 8: Terrain map image from Figure 6, separated into seven base colors using the RGB clustering algorithm.

Using these classifications, we have established a mapping between GIS and data parallel operations which will allow us to perform now systematic parallel implementation of the core primitives. The following list summarizes our results:

- ☐ Local operations → Elementwise parallel operations
  - Common operators → same as sequential
  - Logical operators → elementwise logical operation
  - Reclassification → WHERE, =
  - Selection → WHERE
  - Statistical → a) inter-layer parallel operations or b) corresponding data elements stored in local memory
- ☐ Focal operations → Regular pattern interprocessor data movement
  - Surface functions → Convolution
  - Statistical → a) data movement around window with computation, or b) neighbor values stored in local memory
- ☐ Zonal operations → Selection, Reduction, Scan, Permutation
  - Contiguous/Statistics → a) segmented scan or b) parallel operations on each zone
  - Contiguous/Geometry → reduction
  - Non-contiguous → select each zone sequentially to perform parallel operations
- ☐ Global operations → Scan functions
  - Euclidean distance function → segment scan, selection
  - Weighted distance function → segment scan
  - Visibility/Line of sight → segment scan
  - Grouping → segment scan, distance doubling
  - Expand/shrink → maximum distance neighborhood searching
  - Raster-vector conversion → segment scan
  - Accumulative operation → reduction.

As seen from this summary, most local, focal and zonal GIS functions admit natural and relatively straightforward implementation in terms of standard data parallel operations. For some global operations, algorithm design and implementation is more complex. However, many data parallel algorithms related to global operations already exist or can be adapted from other, better explored domains of parallel processing such as machine vision.

We have recently initiated the process of a trial parallel implementation of selected GIS primitives, aimed at exploring issues such as networking, user interfaces and overall organization of the parallel GIS services. We adopted the public domain raster GIS package GRASS as the tentative user front-end and classification library and we developed the direct socket library based connection between the GRASS host and the CM-5 host. We are currently exploring possible extensions of this work towards higher level communication protocols such as PVM and more powerful integration and visualization front-ends such as offered by AVS.

5. Towards Parallel AVS Environment

The particular AVS based integration strategy described in Section 2 is the most natural when the base volume of the parallel code exists and the integration goal is to provide distributed I/O or pre/post-processing services and/or flexible interactive control of simulation parameters. The most practical approach in such cases, especially for large and complex research codes, is to handle the core part of a parallel application as a single, extensively parametrized AVS module. In general, parallel AVS modules in this type of approach correspond typically to parallel machines available on the network, with each computationally intensive component of an application handled by a dedicated massively parallel system.

On the other hand, new projects aimed at developing a large volume of parallel code from scratch would typically benefit more from the AVS-style modular organization, built at the early design stage into the parallel code itself. The question arises, however, what is in fact the most natural "parallel AVS-type" software development technology. There are no standard solutions yet and also there is clearly no unique answer as the problem is both architecture and application domain dependent. We focus here on distributed memory multiprocessors such as CM-5, iPSC or nCUBE2 and on the data parallel processing domain such as raster GIS support, discussed in the previous Section.

A MIMD-parallel GIS environment could be in principle written as a single, large HPF (High Performance Fortran) code with individual GIS filters mapped on suitable data parallel Fortran routines and with some custom front-end to generate suitable combinations of HPF/GIS library calls in response to requested sets of filters, selected interactively by the map analyst. Such approach, however, would be impractical for large GIS systems, both in the development period (compilation time bottleneck) and in the production period (node memory size limitations, limited interactivity). We see the HPFI (Interpreter) based approach to be more suitable in such computational domains. In the HPFI model, each node of a multicomputer runs a compute server which provides base HPF functionality such as runtime (re)alignment, decomposition and arithmetic for data parallel arrays. Domain specific operators such as GIS filters can be then constructed as suitable HPFI scripts and organized as a HPFI/GIS shell, i.e. an interactive, domain specific HPF language extension.

A natural next step is to view GIS shell commands as AVS-type modules and provide the visual network editing tools for assembling composite GIS filters from existing primitives. We denote by

$$p_i^{(\alpha)},\ i = 1, 2, ..., N,\ \alpha = 1, 2, ..., M \qquad (3)$$

a computational process running on the *i*–th node of an N-processor system and representing this node's component of a data parallel AVS-type module labelled α. The full data parallel module $m^{(\alpha)}$ is then represented by a collection of node processes:

$$m^{(\alpha)} = \left\{ p_1^{(\alpha)}, p_2^{(\alpha)}, ..., p_N^{(\alpha)} \right\} \qquad (4)$$

and the full computational content n_i of the *i*–th node is given by a collection:

$$n_i = \left\{ p_i^{(1)}, p_i^{(2)}, ..., p_i^{(M)} \right\} \qquad (5)$$

Processes communicate in the AVS (dataflow) style in the upper (Greek) indices and in the HPF (collective message passing) style in the lower (Latin) indices. This generic model can be implemented in several different ways and it raises some non-trivial design, communication, mapping and concurrency control issues. For example, the full analogy between "data parallel AVS" and "distributed AVS" approaches requires multitasking operating environment in each node of the multiprocessor – otherwise the processes in eq. (5) would block each other and allow only for simple linear (acyclic) dataflow patterns.

An exploratory project aimed at designing the API interface for parallel module builders and developing the trial library of data parallel modules for transputer systems was recently conducted in the Edinburgh Parallel Computer Centre [TWF93][Faig92]. In this approach the emphasis was on clean separation between (explicit) dataflow communication channels and (implicit) collective message passing channels, whereas mapping issues were handled simply by piling collections of module codes such as in eq. (5) into a single executable per node.

At NPAC, we are currently exploring an alternative approach, following the HPFI concepts mentioned before and using the MOVIE server technology at the virtual machine layer. MOVIE (Multitasking Object-oriented Visual Interactive Environment) is a multithreading interpreter of MovieScript – an object-oriented extension of PostScript with support for index-free (APL, HPF type) matrix algebra [FFHNPS92]. A Virtual Machine model consists of a mesh of MOVIE servers, communicating by "MovieScript passing" in a similar way as Adobe PostScript communicates with the printer or the SUN NeWS PostScript communicates with the window server. HPF scripts are runtime translated to MovieScript and processed by node servers. Each $p^{(\alpha)}{}_i$ is a MovieScript thread, scheduled according to a preemptive, real-time policy which guarantees full concurrency of parallel modules.

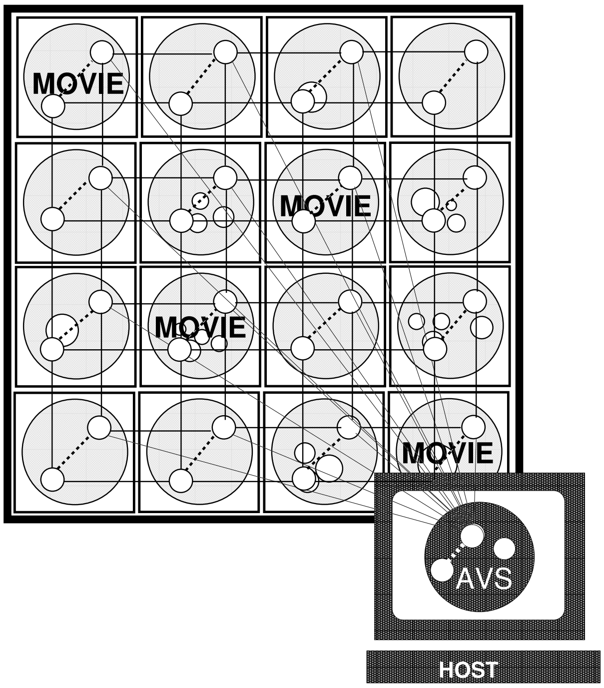


Fig 9: MOVIE based model for parallel AVS with M = 2 modules on a distributed memory multicomputer with N = 16 nodes.

Fig. 9 illustrates the minimal non-trivial AVS-style dataflow network, following the model discussed above. It consists of two (M = 2) data parallel modules: *io* and *compute* on a mesh connected array of N = 4x4 = 16 processors. Threads are represented by white circles within shaded server circles and HPF/AVS-type connectivities are represented by solid/doted lines. Each data parallel thread and their dataflow connections are represented on the host in terms of a suitable visual metaphor, such as for example an AVS-style visual network editor. Due to the preemptive multithreading, some other, irregular parallel modules (e.g. employed for interactive debugging or performance visualization purposes) can coexist with data parallel constructs as illustrated in Fig 9. Since thread creation/annihilation operators are part of MovieScript which in turn is both a computational and communication protocol, all thread ensembles in Fig. 9 are dynamic, transient and mobile.

6. Summary

We have presented recent NPAC results on using and evaluating AVS technology for software integration in HPDC with the focus on the GIS application area. So far, the system has been explored for converting individual large and complex parallel codes into interactive distributed applications (Section 2) and for developing advanced visual processing tools for selected GIS filters (Section 3). We are now pursuing a more systematic and complete analysis of a class of data parallel algorithms, most adequate for high performance GIS services (Section 4) and we are developing a powerful dynamic software model for parallel AVS, integrating functional dataflow and HPF based data parallel programming techniques (Section 5). We see AVS technology as promising in the context of the planned large scale project at NPAC towards the Integrated Decision Support System for Environmental Policy/Analysis (Section 1).

7. References

[BCFHR93] Bozkus, Z., CHoudhary, A., Fox, G.C., Haupt, T. and Ranka, S., "Compiling Fortran90D/HPF for Distributed Memory MIMD Computers", Technical Report SCCS-388, 1993, NPAC, Syracuse University.

[CLFMH93] Cheng, G., Lu, Y., Fox, G.C., Mills, K. and Haupt, T., "An Interactive Remote Visualization Environment for an Electromagnetic Scattering Simulation on a High Performance Computing System", Technical Report, 1993, NPAC, Syracuse University.

[Faig92] Faigle, C., "The NEVIS Prototype Library User's and Developer's Guide", EPCC-SS92–8, 1992, Edinburgh Parallel Computing Centre, University of Edinburgh, Scotland.

[Furm92] Furmanski, W., “MOVIE – Multitasking Object-oriented Visual Interactive Environment”, in Fox, G.C., Messina, P. and Williams, R., editors, *Parallel Computing Works*, Morgan and Kaufman, 1992 (in preparation).

[FFHNPS92] Faigle, C., Furmanski, W., Haupt, T., Niemiec, J., Podgorny, M. and Simoni, D., “MOVIE Model for Open Systems based High Performance Distributed Computing”, in *Proceedings of the First International Symposium on High Performance Distributed Computing, HPDC-1 Sept 9–10, 1992, Syracuse, NY*, IEEE Computer Society Press.

[Li92] Li, B., “Opportunities and Challenges of Parallel Processing in Spatial Data Analysis: Initial Experiments with Data Parallel Map Analysis”, in *Proceedings of GIS/LIS'92*, San Jose, CA, Nov 1992.

[Toml90] Tomlin, C.D., *Geographic Information Systems and Cartographic Modeling*, Prentice Hall, NJ, 1990.

[TWF93] Thornborrow, C., Wilson, J.S. and Faigle, C., “Developing Modular Application Builders to Exploit MIMD Parallel Resources”, Technical Report, EPCC-SS93, 1993, Edinburgh Parallel Computing Centre, University of Edinburgh, Scotland.