5.0 SuperMUSE Parallel Computing Hardware

Evaluating uncertainty and sensitivity can be a difficult task, even for low-order, single-medium environmental models driven by a unique set of site-specific data. Quantitative assessment of integrated, multimedia models that simulate hundreds of sites spanning multiple geographical and ecological regions will ultimately require a comparative approach using several model evaluation techniques, coupled with sufficient computational power. Efforts to address computational needs for 3MRA model evaluation are outlined here.

Discussed in Section 2.5.3, in addition to total uncertainty imparted through model inputs and model error, sampling-based probabilistic analyses are also subject to random sampling error introduced in the output distribution(s) sought (i.e., OSE). This is an aspect of computational constraint in propagating total uncertainty through the model. If the model is large or complex, and overall runtimes needed to accurately describe the output distribution(s) are extensive, then output sampling error, itself not a function of lack of knowledge or data, can become a significant element of uncertainty in the final analysis. The longer an individual deterministic run time is, the greater the influence that output sampling error would have in a probabilistic analysis for a given computational capacity. This constraint affects all sampling-based designs, including both those investigating uncertainty or sensitivity of a modeling system. As a general rule, the more thorough an uncertainty or sensitivity analysis is, the more individual model runs that are needed to establish it, and the greater the computational capacity required.

Presented in Sections 3.3.1 and 4.5, the 3MRA modeling system includes a set of 17 science modules that collectively simulate release, fate and transport, exposure, and risk associated with hazardous waste constituents disposed-of in land-based waste management units (WMU). The 3MRA model currently encompasses 966 input variables, over 185 of which are explicitly stochastic. A characteristic of both uncertainty analysis (UA) and sensitivity analysis (SA) for very high order models (VHOMs), like 3MRA, is their need for significant computational capacity to perform many relatively redundant simulations. While UA/SA is emerging as a critical area for environmental model evaluation, investigating uncertainty and sensitivity in personal computer (PC) based, Windows-based models has been historically limited by a lack of computing capacity. Equally, higher-order UA/SA algorithms (e.g., variance-based global techniques such as FAST or Sobol's Method) warrant study to determine their efficacy in establishing requisite confidence in the use of VHOMs for regulatory decision-making.

Representing a robust solution to the computational dilemma imposed by sampling-based UA/SA for many Windows models, design of **SuperMUSE**, a 225 GHz PC-based, Windows-based **Super**computer for **M**odel **U**ncertainty and **S**ensitivity **E**valuation, is presented.

The discussion in this section focuses primarily on the hardware elements of the SuperMUSE design approach. Supporting software for facilitating use of SuperMUSE in model evaluation strategies (Babendreier and Castleton, 2002) along with key enhancements needed for 3MRA model evaluation embodied in a pending release of 3MRA Version 1.x (Figure 1-1) are further described in Section 6.

5.1 3MRA Model Evaluation Computational Needs

As part of an extensive model evaluation process, uncertainty and sensitivity analyses (UA/SA) are being undertaken by OSW and ORD to evaluate the 3MRA modeling system. These efforts will rely upon the use of tens of millions of 3MRA simulations to test effects of small and large changes in model inputs, and will require a computational capacity not practically achieved through use of one, or even several, desktop PCs. For example, the UA/SA simulation set for a national assessment involving 43 chemicals, 419 site-WMU combinations, 5 waste stream concentrations (C_W's), and 1000 iterations each, represents over 90,000,000 individual model runs. This analysis assumes that the 966 input variables will express sufficient variation and interaction within 1000 iterations to confidently describe the uncertainty and sensitivity of 3MRA model outputs. Assuming an average model run time of 2 minutes, the effort of simulating $90x10^6$ individual 3MRA model runs on a single, modern PC would take over 300 years. Module-level and system-level verification activities can impose equally large requirements for stressing the modeling system over the allowable ranges of inputs needed to initially verify modeling system behavior (e.g., identification of unexpected model errors at runtime, etc.).

Of course, not all 966 inputs of 3MRA are subject to uncertainty and sensitivity manipulations (Table 2-3). Nonetheless, the UA/SA computational effort is significant for 3MRA. While more directed investigations of uncertainty analysis (e.g., a single site, 1 chemical, etc.) become more feasible in stand-alone execution mode (i.e., using a single PC), a full model evaluation effort was recognized as not being readily distinguished as feasible from the outset. For this reason, ORD undertook a concerted effort to deliver supercomputing capacity to Windows-based model evaluation problems, such as that needed for the 3MRA Version 1.0 modeling system.

5.1.1 Massively Parallel Vs. Embarrassingly Parallel Modeling Problems

A fundamental characteristic of UA/SA, particularly for large or complex models, is their need for high levels of computational capacity to perform many relatively redundant computer simulations, where only model inputs change (only relatively slightly) during each simulation. Computational needs for UA/SA represent a fundamental departure from what is commonly referred to as "massively" parallel computing (Brightwell *et al.*, 2000).

Massively parallel computing divides a single simulation into several pieces for distributed execution across 2 or more clients (or nodes or PCs or CPUs). This form of model parallelization can be described as having the characteristic of inter-nodal communication dependency, and presents significant challenges to computational organization for both hardware and software components. Massively parallel computational schemes are a practical tool, for example, in dealing with single scenario, single CPU runtimes on the order of days to decades. An example of this type of problem would be a regional watershed study conducted at a detailed spatial resolution. One major hurdle that massively parallel problems face is the issue of solution scalability. Massively parallel solutions can become inefficient, for example, when the grain-size associated with model deconstruction results in more grains (or submodels per unit computational cycle) than available CPUs. For example, significant inefficiencies are experienced if 28 CPUs are available, but the solution involves 29 parallel operations, each with similar runtimes.

In comparison, the computational problem presented by UA/SA may be practically viewed as retaining a simpler characteristic of inter-nodal communication independency. In juxtaposition, one might refer to the UA/SA problem as "embarrassingly" parallel, since, at first glance, it offers relatively insignificant challenges from the perspective of computer science, which is more squarely focused today on a dominant paradigm of massively parallel computing for highly complex, single-simulation problems. There is typically not much point, for instance, in deconstructing a model for massively parallel operations when the number of full model runs needed is greater than the number of CPUs available. An exception may be when only certain machines can be used to perform specific subtasks (e.g., remote servers with proprietary models or data, etc.).

The term "embarrassing", as introduced here, is meant to draw attention to the distinction that the task of running a model over and over again for purposes of UA/SA is both abject in its relative redundancy, yet simple in its solution. With only a single PC to leverage, there is utter hopelessness in the task for a complex problem statement such as a national site-based risk assessment, yet it remains a task of paramount importance to be completed.

Unlike massively parallel problems, the issue of scalability is not necessarily an inherent problem in UA/SA sampling-based designs using several CPUs, where the number of model runs needed exceeds the number of CPUs available by some integral factor (e.g., 1000 model runs needed where 10 CPUs are available). Here, a single model run serves as an efficient grain-size.

5.1.2 Historic Responses to Embarrassingly Parallel Modeling Problems

When such an embarrassing problem is presented to most UA/SA researchers, the initial inclination is to accept the computational limitation of random sampling designs, and rely instead on approximations of the system (e.g., further simplified model constructions, LHS, etc.). The overall approach represented is to reduce individual model runtimes, as well as the overall number of model runs needed, or both. These approaches can be practical or more efficient in some cases, and should indeed be undertaken if feasible. It becomes increasingly difficult though to justify such an approach at the outset as a matter of course for solving large or complex modeling problems, which may retain characteristics of non-linearity or non-monotonicity, or in which model simplification significantly increases model error.

Another common response to the embarrassingly parallel problem of UA/SA is to turn the problem over to proprietary knowledge bases with expertise in supercomputing, for example national computing centers with either PC-based or mainframe supercomputing infrastructures (the term "mainframe" is used loosely here with respect to current terminology in computing sciences). Such an approach currently relies almost exclusively on operating system paradigms based in Linux or Unix, respectively. Solution in this vein for a Windows-based model requires either complete conversion of the model software to Unix/Linux, or development and maintenance of two versions of the modeling system (e.g., a Windows-based version and a Linux-based version). For complex Windows-based models, or even simple models with many runs needed, this solution approach can be fraught with problems including:

- Initial cost of software system conversion to Linux or Unix,
- Assurances that the two software versions represent the same model,
- Cost of dual-track software development over time,
- Reduced ability to independently verify quality assurance,
- Delays involved in arranging for national computing center assistance, and
- Significant distancing of model developers and users from the process of model evaluation.

Reliance on "supercomputing center" assistance may involve significant annual competition with other large modeling systems for runtime (e.g., competition with massively parallel problems), months to years of planning simulation experiments, and continual budgetary constraints. Should a significant logical error be detected during runtime once the model reaches the actual execution stage in this process, the process may also have to be restarted. This approach and process severely limits the modeler in quickly adapting to knowledge gained during early experiments. While clearly there will always be a complementary role for non-Windows-based supercomputing centers (e.g., massively parallel problems, Unix/Linux models, massive data piping, etc.), the current onus of their use for Windows-based modeling problems can serve as a deterrent to conducting thorough model evaluation.

To reduce the distance of model developers and users from the tasks of UA/SA remains a desirable goal in supporting realistic expectations that model developers and users will thoroughly, and enthusiastically, evaluate models. The Office of Research and Development is continuing to evaluate the benefits of creating innovative hardware and software modeling system solutions that can more efficiently tie researchers, model developers, assessors, and decision-makers directly together in a seamless environment. Such efforts support the Agency's long-standing goal of infusing quality assurance in all aspects of model development and their use in regulatory contexts.

5.1.3 PC-Based Parallel Computing Clusters

While UA/SA is emerging as a critical area for environmental model evaluation, resources to conduct "embarrassingly parallel" computations for Windows-based models have been limited by an associated lack of supercomputing capacity. In addition to the general inaccessibility to technologies that deliver UA/SA capability (e.g., integrated random sampling design in software, advanced SA algorithms, etc.), PC computing capacity has been a driving factor in the typical avoidance by Windows modelers to perform extensive model evaluation.

Increasingly common, particularly for Linux-based systems and massively parallel problems, distributed PC-based supercomputing has expanded rapidly in recent years (e.g., Linux-based Beowulf clusters). Less common though are PC-clusters that support Windows-based models for massively parallel problems or "embarrassingly parallel" problems like UA/SA for 3MRA. Dual-boot systems that facilitate both Linux-based and Windows-based parallel computing are even less common. Also uncommon are associated hardware systems offering dedicated KVM (keyboard, video, mouse) control, which can provide more efficient capabilities in managing many nodes (i.e., PC clients) with relatively unstable operating systems such as Windows. Stability in Windows environs is less and less a problem since the arrival of Windows 2000 and Windows XP, in large part due to better memory management that has reduced memory fragmentation problems in long-running programs.

While it is possible to emulate Windows-based models within Linux OS, three features of this solution approach are:

- There is still the same basic need to solve the embarrassing problem of UA/SA,
- It is not the same modeling environment the user or decision-maker wants to run, or that most stakeholders would be able to run, and
- Extensive compositional validation would still have to be undertaken to ensure precision and accuracy between OS solutions.

The above statements focus on the primary dilemma of familiarity with OS. The idea that it's a Windows-based problem is, on most levels, immaterial. The problem derives simply from the fact that most people wanting to use regulatory models today only have familiarity with Windows operating systems. Evidence of this is found in the overwhelming trend within the Agency to develop Windows-based models for its client base (e.g., Program and Regional Offices, stakeholders, etc.), and the ubiquitous use of Windows OS in most office environments. If most model developers and users were using Linux, the argument would merely shift to the OS of the day, and the "embarrassingly parallel" problem would still need to be solved. In the latter case with Linux, a richer set of software tools and hardware configurations are more readily available. The latter aspect, however, has not brought about a rush by the masses of model developers and users to convert to Linux OS, and deploy their models on associated Beowulf clusters in order to perform UA/SA.

5.2 ORD's Supercomputer for Model Uncertainty and Sensitivity Evaluation

To facilitate UA/SA model evaluation tasks for EPA's PC-based modeling systems, which use Windows operating systems extensively, ORD recently developed a 225-GHz

Supercomputer for Model Uncertainty and Sensitivity Evaluation (**SuperMUSE**). This approach was matched with companion efforts to develop a supporting software infrastructure to conduct "embarrassingly parallel" computations. Designed and constructed at NERL's Ecosystems Research Division (ERD) in Athens, Georgia, the current system is shown in Figure 5-1.

The SuperMUSE development approach has sought to support, in theory, the capability for dual-boot operation, supporting either Windows-based or Linux-based modeling systems. The initial effort focused on Windows-based operations for 3MRA. SuperMUSE supporting management software has been strategically designed in Java, a platform independent programming language (i.e., independent of PC operating system choice), to deliver a Linux-based capability simultaneously.

5.2.1 SuperMUSE Initial Design, Cost, and Effort

Major components of the existing SuperMUSE (Figure 5-2) include a front-end program server, a back-end data server, and 176 client PCs, each with a minimum of 256 MB RAM. A variety of Windows operating systems are supported (i.e., Windows 95, 98, NT 4.0, 2000, etc.). Interconnections were achieved through use of 16-port Raritan KVM switches, and 24-port Linksys (10/100) network switches, the latter branching-up to a master CISCO 3550-24/2 enterprise network switch. The system network communication protocol is based on TCP/IP. The system design currently provides for gigE channel (1000 megabits/sec) data flow to and from servers, and allows for single-user KVM remote access. A picture of the actual network and KVM switches in use is shown in Figure 5-3. Example cabling layout is shown in Figure 5-4, where the blue category 5 (i.e., Cat5, Cat5e) cabling is used for network switching, and the larger white cabling for KVM connections.

Various combinations in the cluster design are easily achieved, and depend on the financial resources available (e.g., client speeds, server storage capacities, etc.). Representing a capacity to support 192 clients, the initial SuperMUSE switch layout (with 121 actual client PCs) was acquired for $125,000 in 2001. This cost excludes servers and 16 older 333 to 450 MHz processors initially used to seed the project. Optimal purchasing based on $/GHz for client PCs will typically identify 3 to 6 month-old CPU technology. Current processors in SuperMUSE range from 333MHz to 2.3GHz. Typical PC costs to fill-out the remaining capacity of the switch system are on the order of $900 per PC. One of the great potentials of the SuperMUSE scheme is the exploitation of older PCs no longer in use, or in the eventual use of grid-based computing configurations to leverage dormant PC-based computational cycles within an organization, or across organizations (e.g., SETI at Home).

Electrical Service and Heat Dissipation

Pre-design considerations include available space, and room heating and cooling requirements versus capacities. Plated wire shelving is best for ease of cabling and heat dissipation. The 192-PC shelf-system shown in Figure 5-1a is currently supplied by a 100 amp three-phase electrical service. An important aspect in specifications for new PCs was the acquisition of "small-form" factor designs, which use 100 watt (W) power supplies for Pentium-3 chips and 160W supplies for Pentium-4 chips. This is compared to standard, larger form factors, such as typical workstation units found at your desk that normally use 200W and 330W power supplies, respectively. Heat dissipation issues are typically equivalent between small-form and large-form factor designs, owing to the fact that the CPUs are the primary source of heat output in a single PC.

The cost of initial SuperMUSE hardware and software development was relatively small compared to costs that would be incurred in 3MRA software conversion to Linux/Unix alone. A key cost-saving multiplier of this approach is that the initial costs of SuperMUSE hardware and supporting software development can also be leveraged for use in evaluating other Windows-based (or Linux-based) models, without additional individual model or modeling system OS conversion costs.

The fundamental philosophy in the SuperMUSE design approach was fitting supercomputing needs with existing trends in model development, as opposed to fitting models to existing trends in supercomputer development.

KVM (Keyboard, Video, Mouse) Switches

KVM switches allow the user to view the desktops of individual PCs in the system from a single terminal (or several remote terminals), which is a boon in assessing runtime environment failures and performing various quality assurance checks during runtime. While it is possible to view individual PC desktops through server software and network adapter cards in the PCs (i.e., through NICs and without KVM switches), any failure in the communication system between PCs necessitates physical removal of the PC to a bench for further evaluation (i.e., the “crash cart” approach).

For example, if certain failures occur (e.g., hard-disk problems, NIC failure/conflicts, model lock-up, etc.), one can only immediately determine that an errant PC failed to boot or crashed after boot, or has otherwise failed to return, but not the reason for failure. Such failures are usually easily diagnosed with system messaging sent to the PC’s desktop screen, accessed through KVM switches. Less expensive solutions, forgoing KVM switches, are manageable, slightly less expensive overall, and far easier to wire-up, requiring only Cat5 cabling between NICS and network switches. A non-KVM switch approach is increasingly more feasible as operating systems become increasingly more stable. The capital investment for KVM switches can range from $20 to $120 per PC client, where this must be balanced against long-term labor and project delay costs incurred in forgoing this approach.

For Windows environs, simple “software KVM” approaches may require use of Windows “server” licensing schemes, which the SuperMUSE design approach purposefully avoided. A notable feature is that one does not need to purchase Windows Server software or client “CALs” to implement the SuperMUSE scheme if the model itself under evaluation does not require this licensing. Such licensing would, for example, be needed to create multiple client connections or mappings to a centralized Access database server for certain model applications. One advantage in accepting the cost of Windows Server licensing is the increased capability to manage system security, which becomes more important with multiple users in the system.

SuperMUSE-type designs should be cognizant that not all KVM switching systems perform equally. In larger systems, many problems can be avoided if the switch design provides for unique emulation circuitry per KVM channel (i.e., a unique emulation circuit is designated for each PC in the system). Each KVM switch in SuperMUSE offers 16 separate emulation circuits (i.e., 1 PC hookup per channel).

5.2.2 Runtime Management Issues

Only servers and the master CISCO network switch currently use uninterrupted power supplies (UPS). The integral software design employed for managing SuperMUSE is capable of handling system level power failures with minimal loss of information for the given experiment(s) underway. Such power failures are relatively infrequent, but can invalidate long running experiments, should they occur. In SuperMUSE, a system-level power failure in the middle of a month-long experiment would result in only the current set of model runs underway having to be restarted (e.g., for 3MRA, an average loss of 2 minutes of computation time per PC). The SuperMUSE software system automatically handles resumption of the experiment as power is restored, without user intercession.

A major consideration in construction and subsequent hardware management is human capital. The effort required for hardware construction is relatively minor; SuperMUSE was easily wired together once power was supplied to the shelving units. SuperMUSE uses custom-length Cat5 cabling made on-site, but off the shelf Cat5 patch cables would also work just as easily. The Linksys network switches used here offer a push-button switch on Channel 1 for "cross-over" to the master CISCO switch, where all Cat5 cabling used can be "straight-through". If you can plug in a PC, you can build SuperMUSE. Subsequent PC hardware management is most effectively dealt-with using typically available service contracts from PC suppliers that are extremely cheap, especially when negotiating bulk purchases of PCs.

The failure rate of SuperMUSE PC hardware (roughly 1 PC hardware failure every three to four weeks) appears to be slightly higher than expected, and may be due to their grueling work schedule, which is basically non-stop, 24 hours/day. The failure rate decreased rapidly after the first year of operation per PC. One issue not expected was the occurrence of several intermittent RAM memory failures observed on Windows 98 units (half of all hardware failures were related to bad RAM memory). These intermittent type failures may be prevalent on all PCs, but have only been noticeable to date for PCs running Windows 98. Unrelated, PCs running Windows 98 typically need to be rebooted on a weekly basis to mitigate (presumably) memory fragmentation problems. "Soft" rebooting of the entire PC client group is easily handled by the SuperMUSE software management scheme with a single mouse click. Windows 2000 OS has shown itself to be extremely stable in the SuperMUSE computing environment; Windows NT 4.0 has been the least workable.

Virus Protection and PC-Client Hard-Disk Image Ghosting

SuperMUSE currently employs virus protection and hard-disk ghosting using Norton software products, which have worked well. Homogeneity across PC clients in both hardware and OS provides simpler management, but it is not a necessity. The machine used for ghosting is an older, low-end PC (shown by itself on the last shelf in Figure 5-1b). Currently, SuperMUSE is physically separate from EPA's local and wide area networks, to avoid the onus of increased security demands associated with such connectivity. However, the system can easily be connected with a single cable, if desired. Overhead costs associated with ERD's LAN/WAN security issues currently outweigh any advantages, for the time being, in establishing such connectivity. The system also uses a DLT-IV back up tape drive system to secure the larger volumes of data on the backend servers (Figure 5-2).

The ghosting approach is used primarily to secure OS and model application configurations on each representative PC-class in the system (i.e., a single PC and image is used to back-up a set of PCs with similar hardware configurations). Currently, SuperMUSE maintains 7 such classes for various PC clients in use. For example, if a PC client' s hard-drive fails, the roughly 3 GB (gigabyte) ghosted software image containing the PC's OS and the 3MRA model application files can be restored onto a newly installed hard-drive in less than 6 minutes, rebooted, and put back to work. Any locally stored data for the experiment (i.e., ELP1 output data for model runs not yet collected; see Section 6) would, of course, need to be re-simulated.

5.2.3 SuperMUSE Expansion and Life-Cycle

Representing a desired maximum capacity, contingent on capital funding resources available, SuperMUSE at ERD will be expanded in the near future to 384 client nodes, plus addition of supporting servers for multiple experimentation, totaling approximately 1000 GHz of computing capacity based on today's CPU speeds. Multi-user KVM access would also be implemented in such an expansion, further complementing an existing SuperMUSE software capability that allows multiple experiments (i.e., multiple model evaluations) to be simulated concurrently. Different models, or different experiments upon a single model can be concurrently evaluated on SuperMUSE. Without the more expensive "multi-user" KVM access switching systems, only a single user can monitor individual PC-client desktops at any one time, though this can be done from several local or remote terminals in a "single-user" KVM system.

The hardware design approach for NERL-ERD's SuperMUSE seeks a feasible cycle of PC upgrading based on a 3 to 4 year PC life cycle, while maintaining a system machine speed 100 to 1000 times faster than the standard desktop PC typically available to model developers and users at any given time. A general trend dealt-with in SuperMUSE-type designs, for the time being anyway, is that model developers typically maintain a relatively constant tolerance for runtime, over time, for complex problem statements. As PCs become faster, model developers often leverage this speed with more complex solutions of a given problem statement, offsetting the gains in CPU speed. This is referred to here as a "developer runtime tolerance" for individual model runtimes, which tends to be more constant than not for a given model developer over a period of several years. This is, of course, a gross generalization; the term "constant" is used loosely here. Nonetheless, the phenomenon is apparent. The result is that the computational burden of UA/SA will tend to remain constant for models and modeling systems over time, despite predictable gains in CPU speed over the same period.

Important to the hopeful exploitation of this ORD effort, a PC cluster can be scaled to any user's needs (i.e., 2 to 1000^+ PCs), and can be constructed and configured by relative novices using off-the-shelf hardware technology and familiar Windows operating systems. The approach in hardware and software design was envisioned to foster development of similar SuperMUSE capabilities across the Agency to fit individual researcher's or programmatic needs. Together with FRAMES 3MRA Version 2.0 (see Section 6.8 and Figure 1-1), and supporting Java software code developed for the SuperMUSE concept, this "commodity-based" solution for "embarrassingly parallel" problems for Windows-based models becomes easier to reach for a wide array of model developers and users. Discussed in Section 6.2, a given model can go directly into SuperMUSE implementation, exclusive of FRAMES 2.0, with the separate creation of a model-dependent "Model Tasker".

5.2.4 MiniMUSE – Staging Software Enhancements for SuperMUSE

SuperMUSE is currently configured into two groups of PCs, a core set of SuperMUSE PCs, and a smaller set of PCs referred to as MiniMUSE, shown in Figure 5-5. This smaller set of 8 PCs, with companion servers, was initially segregated to allow for staging software modifications (both changes to SuperMUSE management software, and changes in models being evaluated), while allowing concurrent experimentation on SuperMUSE. MiniMUSE is used, for example, to work out the wrinkles in the application of a new model or modeling system.

More recently, the older set of 16 (333 to 450 MHz) PCs in the main lab (Figure 5-1b) have been allocated to MiniMUSE due to their small hard-drive sizes (4 to 6 GB) and the storage demands for post-processing (i.e., ELP) schemes being employed in 3MRA for longer experiments (i.e., > 25 national realizations). Currently, SuperMUSE "proper" employs 152 1^+ GHz client PCs with 10^+ GB hard-drives, and MiniMUSE employs 24 (333 to 450 MHz) PCs. This new alignment has been helpful in managing ongoing experimentation while facilitating continual advances in software design, for example, in the ongoing development and testing of components of 3MRA Version 1.x, discussed in Section 6.

The segregation of MiniMUSE network connectivity is not physical, even though portions of it are housed in a separate room adjacent to the main SuperMUSE lab. The entire set of PCs (SuperMUSE and MiniMUSE clients and servers) is actually interconnected using the same communication network backbone. Described in Section 6.2.1, segregation of SuperMUSE and MiniMUSE is effected through use of two "CPU Allocators" running concurrently on two different front-end servers. A simple software switch in every PC client's boot-up file is set to assign a given PC client to any CPU Allocator currently in use. The entire SuperMUSE system can be reconfigured to segregate several PC groups, or to combine all clients into a single PC group, in a matter of minutes.

5.3 Identification of Software Toolset Needed for SuperMUSE

To exploit capabilities of the SuperMUSE parallel computing environment, several software tools were needed. Key functionalities required that were identified included:

- Facilitating the distribution of workloads among PCs,

- Managing files and data across PCs,
- Facilitating model-specific data analysis tasks, and
- Extension of alternative model input sampling tools.

Together, the items of key software functionality requirements noted are grouped, as such, into a pending release of 3MRA Version 1.x. Alternate sampling tools (e.g., LHS, etc.) are not currently available in 3MRA Version 1.x, but are to be incorporated in future software releases (i.e., FRAMES 2.0, etc.). The categories of software development supporting the SuperMUSE hardware design are described in further detail in Section 6.

Three key points are: (1) the combined SuperMUSE hardware and software management components that support parallelization of models are amenable to facilitating model evaluation needs for all PC-based models (Windows and Linux); (2) the 3MRA Version 1.x model-specific enhancements are crucial to implementing the planned UA/SA experimentation for 3MRA; and (3) SuperMUSE hardware and management software, together with 3MRA Version 1.x, provide for a powerful capability to perform millions of 3MRA model simulations per month.

5.4 Benefits of Integrated SuperMUSE Hardware and Software

The following materials summarize key points underlying the motivation behind the SuperMUSE hardware (Section 5) and software (Section 6) design approach described herein for evaluating PC-based environmental models, and, in particular, those characterized as very high order models (VHOMs).

- **Why is Model Evaluation Becoming Increasingly Important?**
    - We need to better communicate prediction uncertainty to decision-makers.
    - We need to improve identification of critical gaps in knowledge and data.
    - We need to address the increasing technical focus in regulatory-driven litigation.
    - Because we are increasingly called upon to establish the validity, trustworthiness, and relevance in model predictions. (Chen and Beck, 1999)

- **Uncertainty and Sensitivity Analysis for VHOMs – How Is It Done Today?**
    - Many techniques and methods are available, and are improving constantly.
    - Current knowledge and execution capabilities are usually limited to a select few, and are often out of reach from most model developers and model users.
    - UA/SA represents an “embarrassingly parallel” computational problem; solutions involve running a model over and over with slightly different inputs.
    - Thorough PC-based model evaluation is often avoided due to a lack of computational capacity.
    - Many EPA models are written for Windows OS, but most supercomputing solutions today require mainframes or Linux-based PC supercomputing clusters.

- **Key Aspects of the UA/SA Runtime Problem**
  - As model complexity, time and space grid density, or types of uncertainty and sensitivity analyzed increases, computational burden (i.e., overall model runtime) typically increases geometrically.
  - A primary reason UA/SA techniques are not widely applied to EPA models today is the lack of Windows-based computational processing capacity.
  - As a general trend, it is typical to see PC-based model developers increase model complexity over time, offsetting concurrent gains in CPU speed.
  - Depending on the EPA model/application, one may need 100's to 100's of millions of model simulations.

Associated benefits of the integrated SuperMUSE hardware and software design are directly applicable to the task at hand of evaluating 3MRA, and are readily extendable to similar tasking for other PC-based models and modeling systems. Figure 5-6 summarizes the benefits of SuperMUSE type approaches in supporting model evaluation tasking for PC-Based VHOMs. Common tasking supported generally includes predictive uncertainty analysis, sensitivity analysis, parameter estimation (PE), and model verification and performance validation efforts.

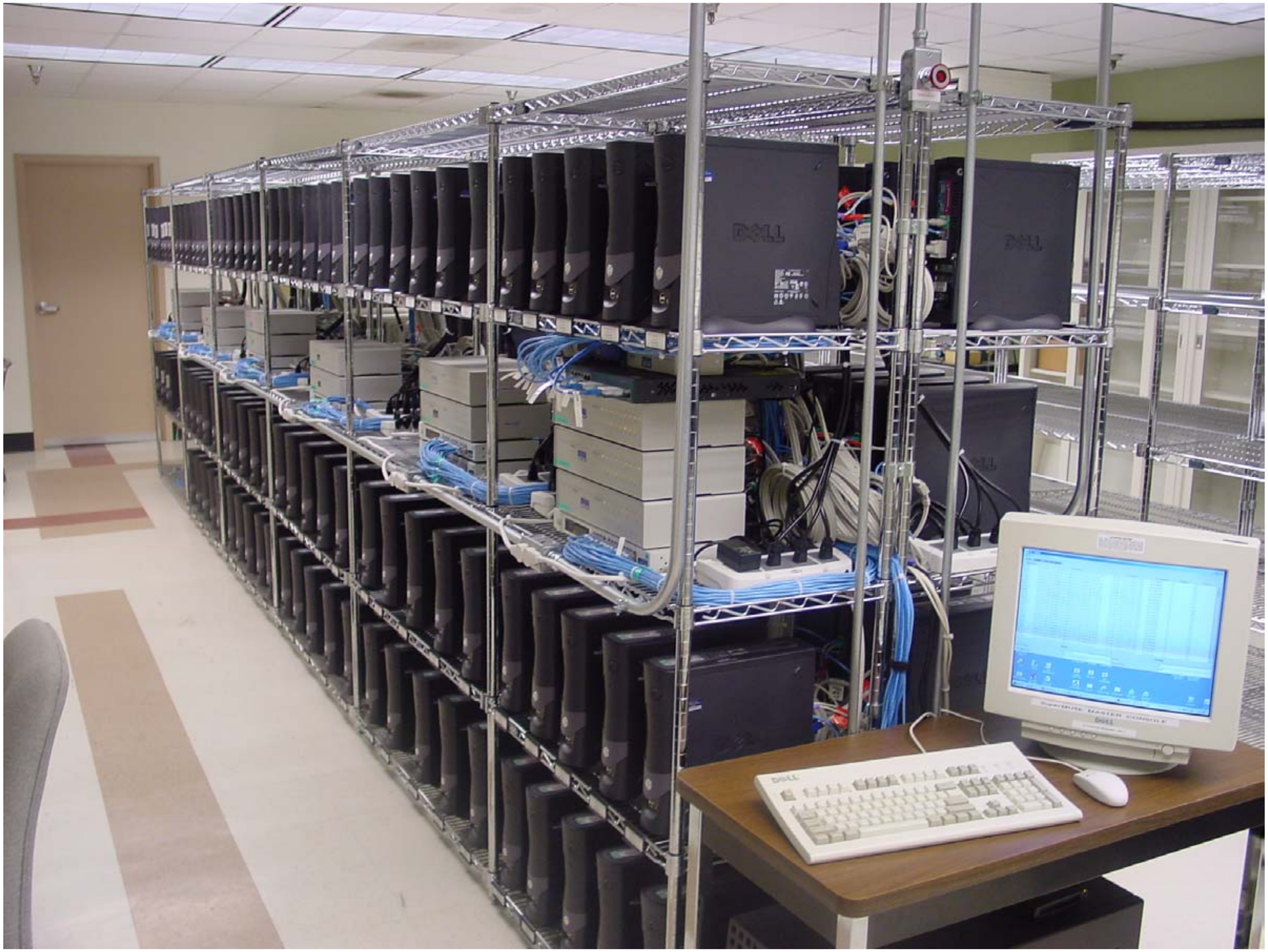


Figure 5-1a. SuperMUSE Parallel Computing Cluster at ORD/NERL/ERD.

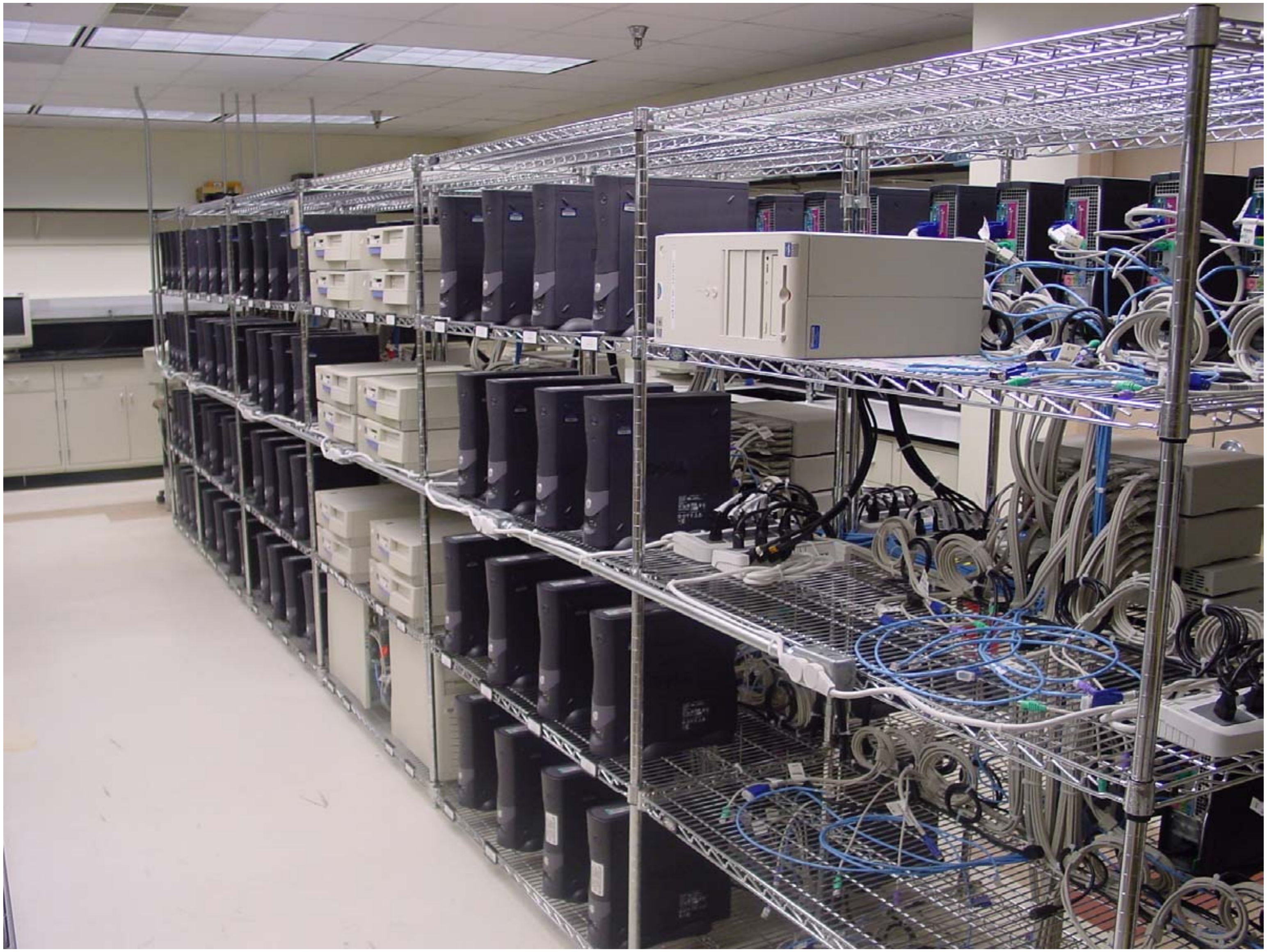


Figure 5-1b. SuperMUSE Parallel Computing Cluster at ORD/NERL/ERD.

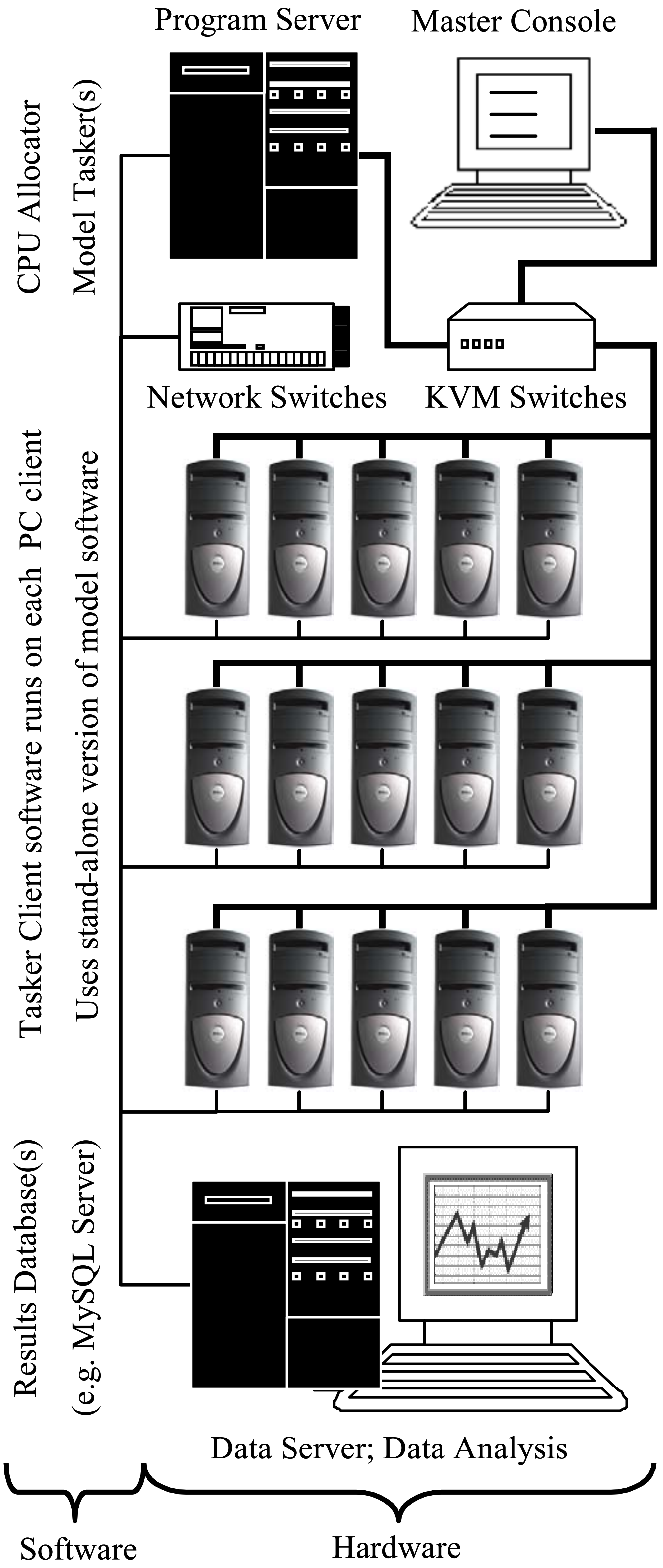


Figure 5-2. Conceptual Layout for PC-Based SuperMUSE Parallel Computing Cluster.

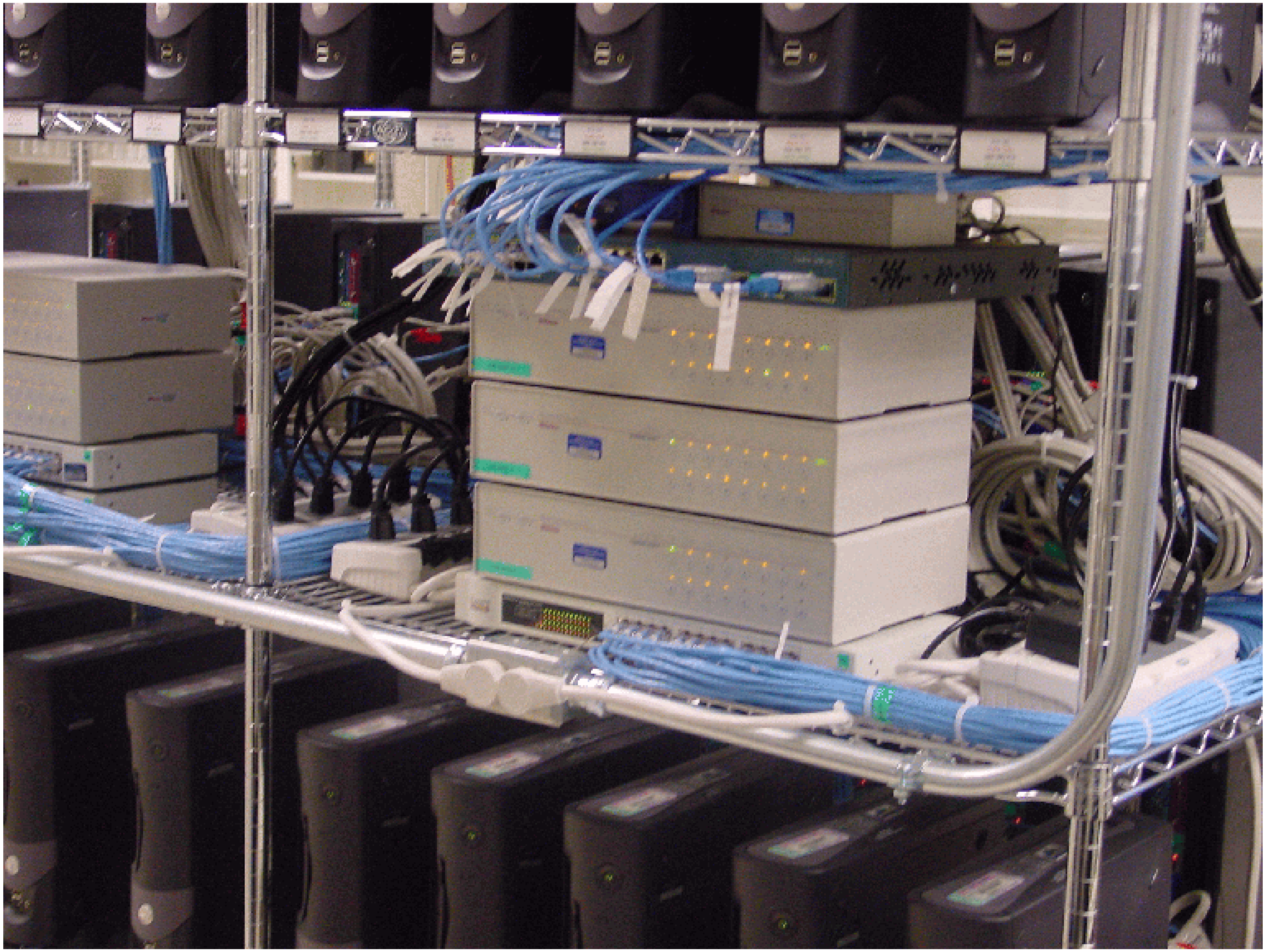


Figure 5-3. SuperMUSE Network and KVM Switching.

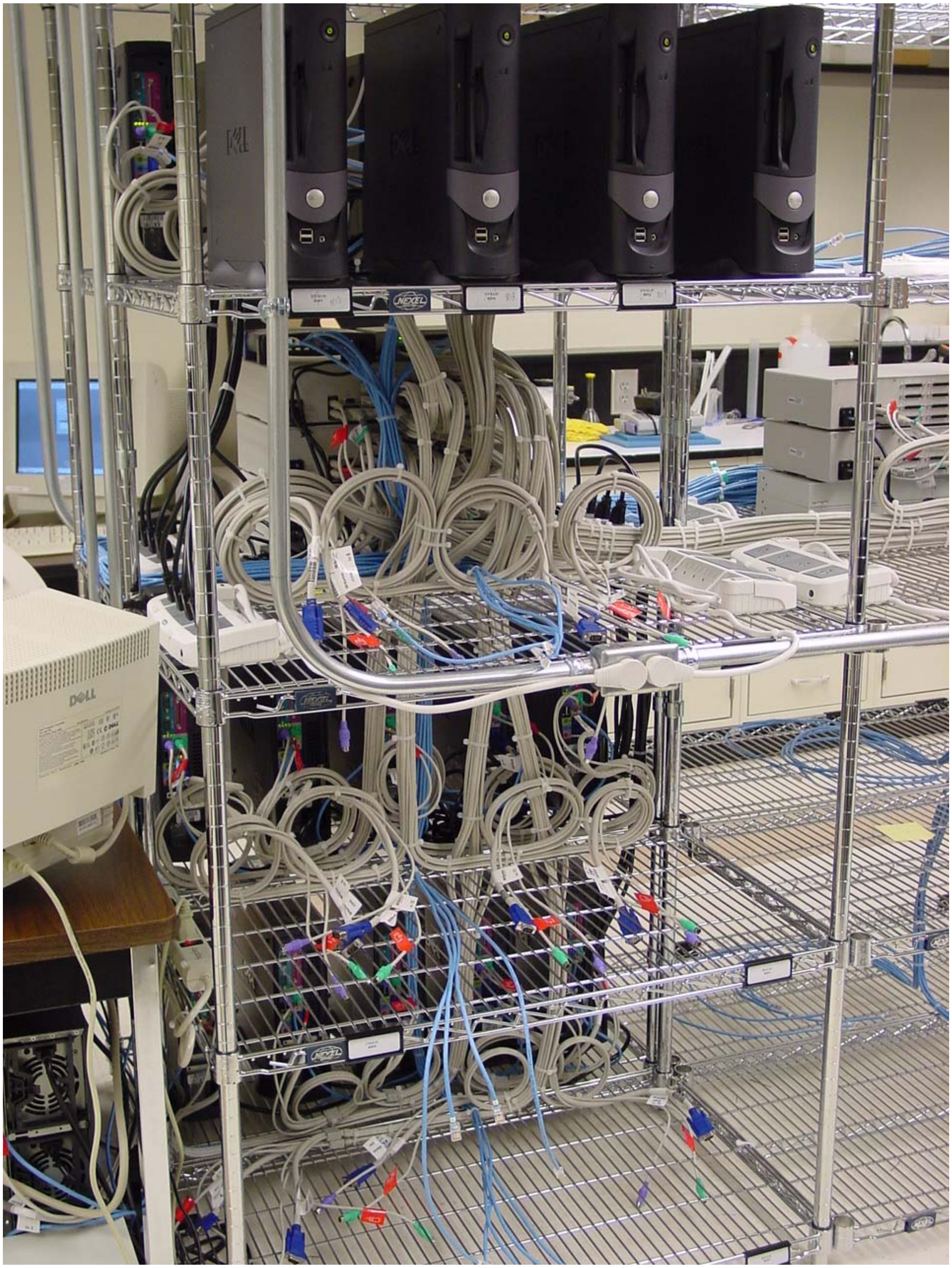


Figure 5-4. Example SuperMUSE Cabling.

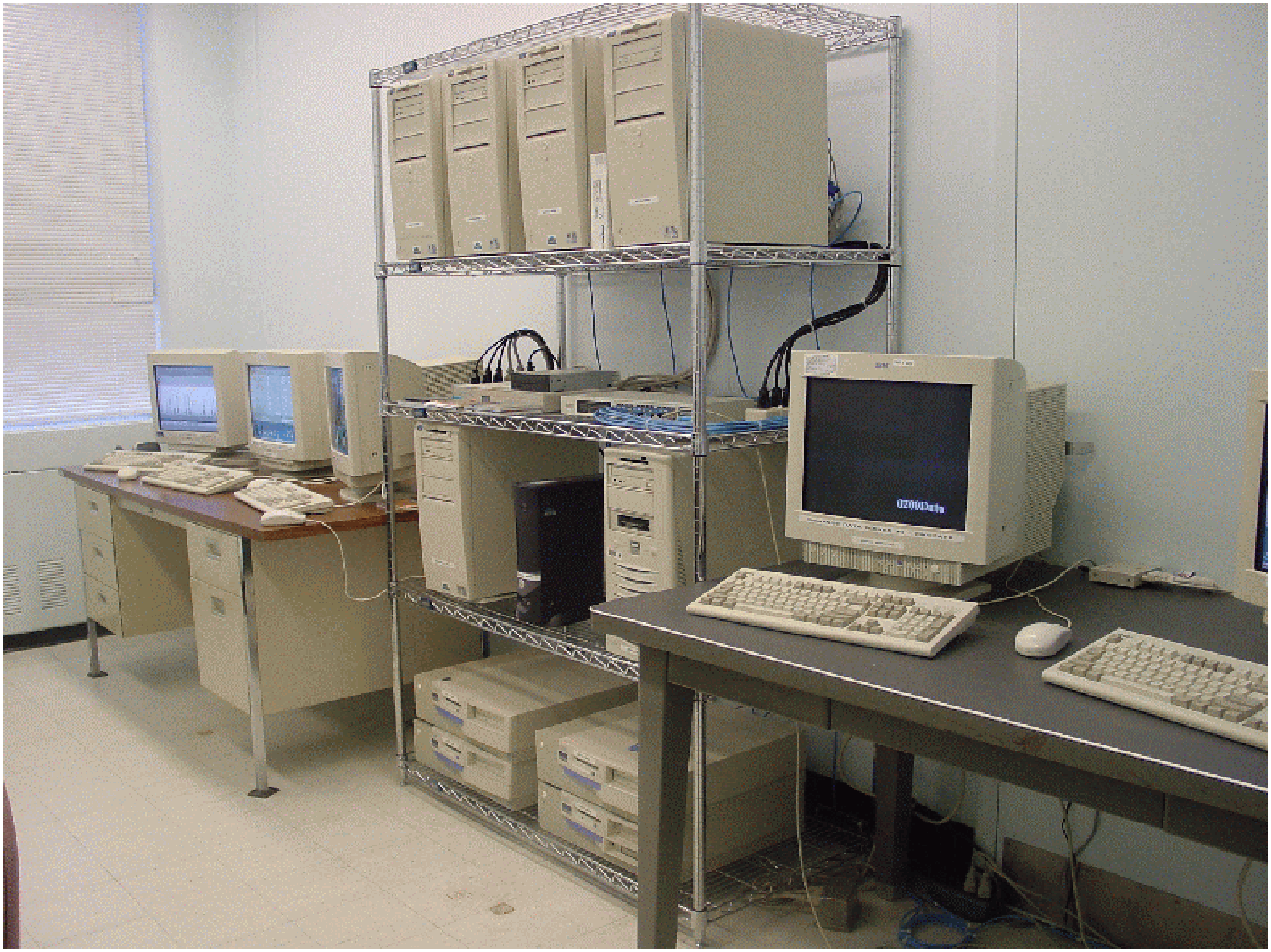
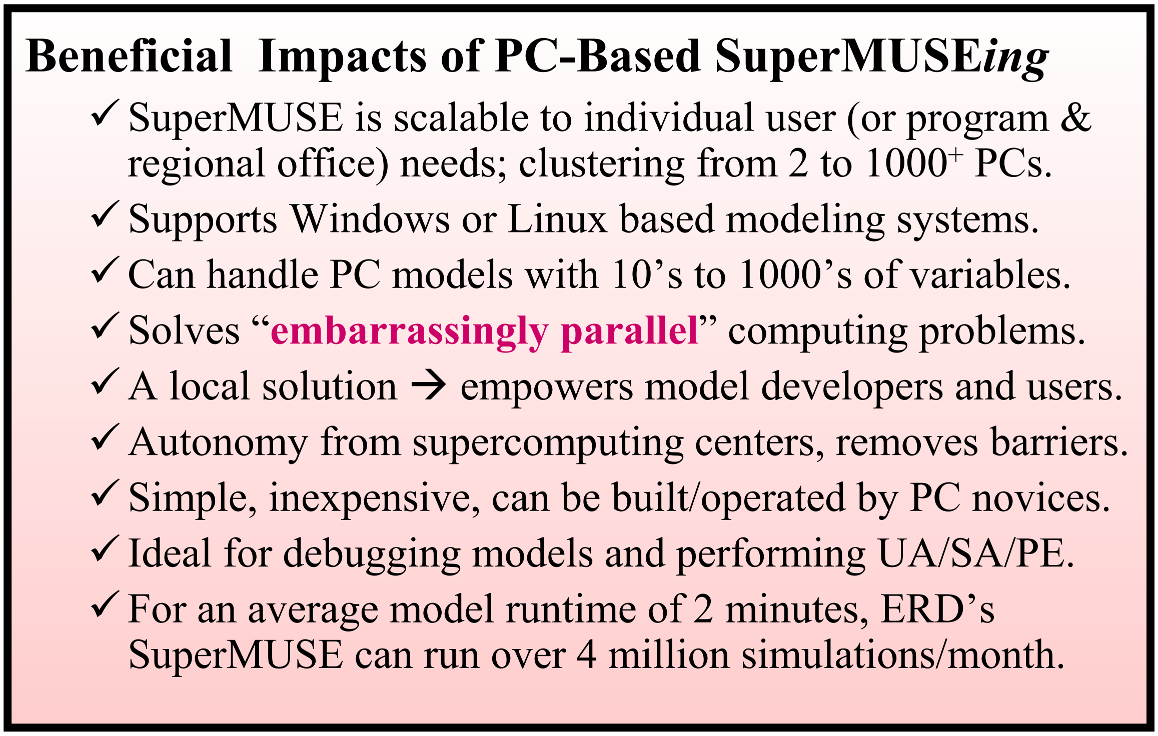


Figure 5-5. MiniMUSE Parallel Computing Cluster at ORD/NERL/ERD.



Beneficial Impacts of PC-Based SuperMUSE*ing*

- ✓ SuperMUSE is scalable to individual user (or program & regional office) needs; clustering from 2 to 1000^+ PCs.
- ✓ Supports Windows or Linux based modeling systems.
- ✓ Can handle PC models with 10's to 1000's of variables.
- ✓ Solves "**embarrassingly parallel**" computing problems.
- ✓ A local solution → empowers model developers and users.
- ✓ Autonomy from supercomputing centers, removes barriers.
- ✓ Simple, inexpensive, can be built/operated by PC novices.
- ✓ Ideal for debugging models and performing UA/SA/PE.
- ✓ For an average model runtime of 2 minutes, ERD's SuperMUSE can run over 4 million simulations/month.

Figure 5-6. Benefits of the SuperMUSE Hardware and Parallel Management Software Scheme.