VERTEX: A Highly Scalable Software Platform for Commodity, Hybrid, Multicore HPC Clusters

Gregory P. Rodgers, Sandhya Dwarkadas*, Ashwini Nanda
greg@hpclinks.com, sandhya@hpclinks.com, ashwini@hpclinks.com (+91 9818 012 576)

July 2011
HPC Links, USA/India
www.hpclinks.com

* Prof. Sandhya Dwarkadas is a Professor at the CS Dept. of Univ. of Rochester and a Distinguished Scientist at HPC Links.

1. Pain Points in Today's Software Platforms for Commodity, Hybrid HPC Clusters

Scalability: As commodity hybrid multicore cluster systems take the front stage in HPC, the number of compute units, or cores, an application has to manage, grows significantly to tens of thousands, or hundreds of thousands. The hybrid HPC systems which use some combination of commodity x86, GPU and Cell BE processors today dominate the top-10 supercomputer list in the world, and are testimony to this trend. As Figure 1 illustrates, the general trend is that as reliability, availability and IO rate increase, system complexity goes up which results in a decline in scalability.

A significant hurdle in achieving scalability in hybrid HPC clusters that use special purpose commodity processors such as NVIDIA GPUs or IBM Cell BE, is that these processors are highly optimized for numeric and scientific computing, but not for integer and branch processing. As a result they are not suited for running a full function operating system when used as a node in a hybrid system. Today's cluster software platforms deal with this inadequacy either by using the GPU and Cell BE nodes as hardware accelerators whereby they are completely dependent on the host processor for all their IO, communication and core management functions, or by letting them inefficiently run heavyweight operating systems. Both these approaches result in loss of usability, flexibility and performance.

Some proprietary software architectures such as IBM Blue Gene deal with the scalability issue by taking a hierarchical approach where IO and control nodes run heavy weight operating systems and take charge of bulk of IO, communication and scheduling functions, whereas the compute nodes run a light weight OS, but still maintain some autonomy over how they communicate among each other and manage their processing cores. Unfortunately, proprietary software packages like those on Blue Gene do not work on today's dominant class of HPC clusters that are hybrid systems based on commodity CPU's, GPU's or Cell BE processors.

Usability: The existing HPC cluster software architectures, including IBM Blue Gene, expose run time management of platform heterogeneity, or hybrid nodes, to the end user. This causes serious usability problems for systems that deploy variety of instruction set architectures and processors in a single, large platform environment.

Flexibility: HPC applications most often need a number of compute nodes working closely together as a single unit to solve a chunk of the problem. The number and types of nodes of this single unit, or 'super node' varies from application to application and needs to remain flexible. Today's architectures either hardwire the ratio and number of various compute and IO nodes, or leave it to the user to handle the details of creating a flexible super node.

2. The VERTEX Solution from HPC Links

The VERTEX (Virtual Execution of Resource Tagged EXecutables) software architecture from HPC Research is aimed at resolving the aforementioned usability, flexibility and scalability issues for commodity hybrid clusters. The current VERTEX platform is a Beta level product from HPC Research being tested through pilots with several customers in the US, Europe and India on hybrid clusters that use combinations of x86, NVIDIA GPU and IBM Cell processors.

The key to the ability of VERTEX solving these issues is a seamless virtual execution environment that makes compute node execution transparent to a parallel application which the user sees as running on a set of control nodes (logical super nodes), or VERTEX nodes. VERTEX addresses the problem of performance **scalability** by providing a hierarchical architecture with distributed VERTEX nodes managing light weight compute nodes on a scalable network fabric. Each VERTEX node is responsible for scheduling jobs and managing IO for a group of compute nodes. Virtual execution is achieved by tagging the executables with resource requirements, allowing fast effective scheduling while supporting standard programming models such as MPI and OpenMP.

Traditional clusters have application-free service nodes for systems management. VERTEX combines service nodes and Blue-Gene-like control and IO nodes to provide a VERTEX node where applications run virtually. Transparent virtual execution is achieved using a lightweight control and IO protocol, 9P. Applications and the application environment appear to run on the VERTEX node, but actual compute intensive binaries are launched on lightweight compute nodes. This paradigm provides a usable and scalable environment from personal supercomputers to exascale-class clusters of super-nodes. Compatibility and portability are achieved using the standardized 9P interface for user-kernel communication. The light weight compute nodes provide a low jitter environment for applications further enhancing system performance and scalability.

System **usability** is enhanced through transparent virtual execution by tagging executables with resource requirements such as type of compute node etc. The virtual execution environment makes compute node execution transparent to applications running on control nodes (VERTEX nodes) by using the Linux binfmt_misc facility. This feature selectively invokes the VERTEX loader for the resource tagged executable (identified as a different binary type).

The transparent virtual execution capability also adds **flexibility** to the size and constitution of a virtual super node, or a VERTEX node, by associating a compute node anywhere in the system to a given VERTEX node. The number and combination of various types of compute nodes controlled by a VERTEX node can adapt to the application requirements at run time.

3. The VERTEX Architecture

HPC architectures support scalable programming models such as MPI by using control nodes for IO and all other system services and leaving compute nodes dedicated to the application. The VERTEX (Virtual Execution of Resource Tagged Executables) architecture implements this model using commodity components. Figure 1 shows the VERTEX architecture.

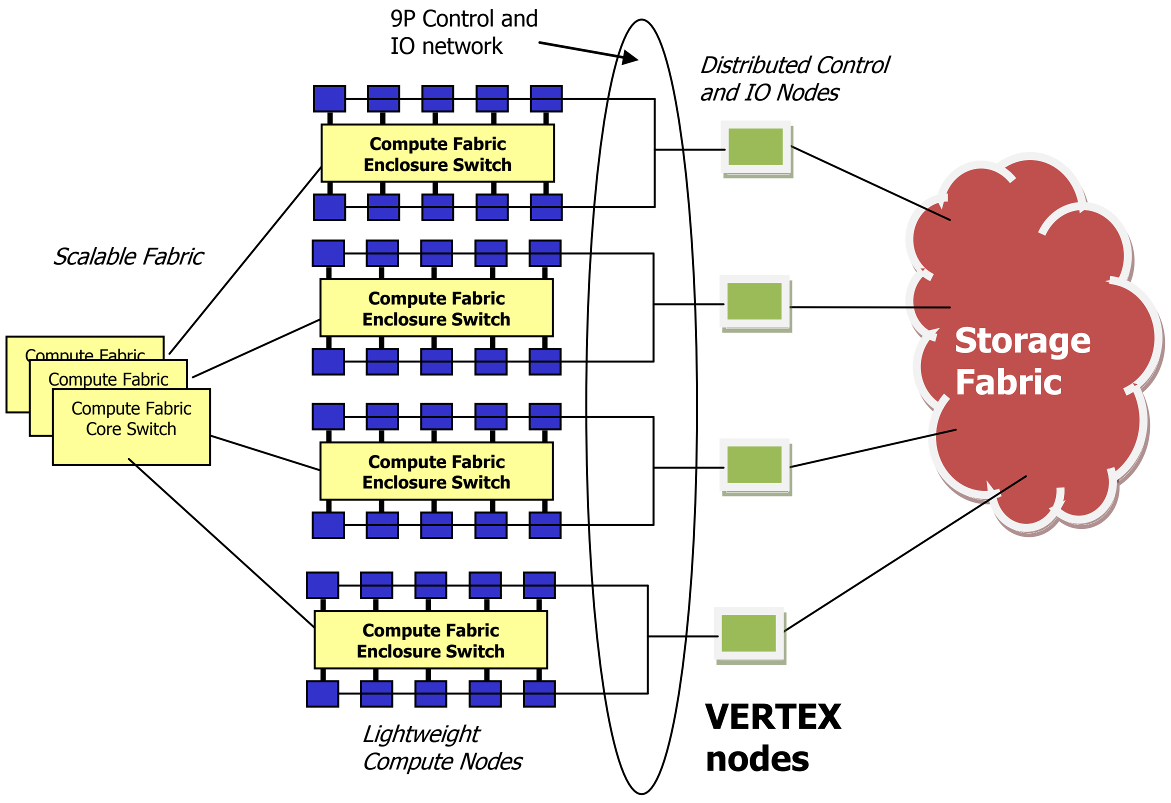

Figure 1 – The Scalable VERTEX architecture.

To the user, this looks like a cluster of high capacity super-nodes. To the application, it looks like an order of magnitude more lightweight compute nodes all connected on a scalable fabric.

We achieve efficiency and scalability with VERTEX in an open environment due to the following enhancements and design choices.

1. We use a single and open protocol (9P) for both execution control and file system IO proxy. This 9P protocol was created by Bell Labs for the Plan9 operating system and subsequently released as open source and incorporated into Linux. Our use of open protocols and open source software enables the VERTEX architecture to be implemented on commodity components such as the SuperMicro SuperBlade (or, HPC Links' SuperLinks) systems as shown in Figure 2.

2. We enable transparent virtual execution by tagging executables with resource requirements. The virtual execution environment makes compute node execution transparent to applications running on control nodes by using the Linux binfmt_misc facility. This feature selectively invokes the VERTEX loader for the resource tagged executable (identified as a different binary type).

3. The VERTEX loader interacts with the vertex scheduler to schedule and start execution on remote nodes via 9P, while making execution appear native to the VERTEX control node. Heterogeneous resources such as CPU and GPGPU cores are scheduled according to the application's tagged requirements. This allows transparent execution of GPU applications on non-GPU nodes with potentially complex IO capabilities.

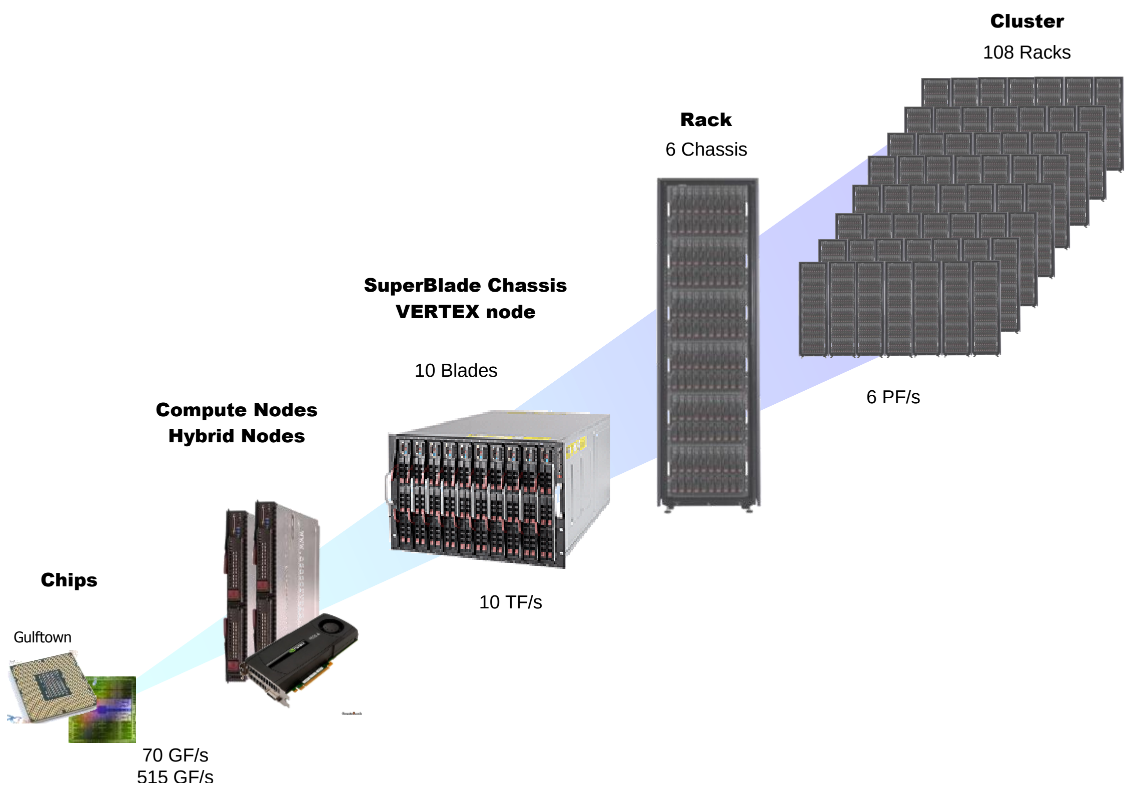

Figure 2 HPC Links' *SuperLinks* family of HPC systems using commodity components: VERTEX is designed for systems like *SuperLinks*

4. VERTEX Software Components

A number of new software components are introduced to implement VERTEX. These are illustrated in Figure 3. These are the five major software components of VERTEX.

1. An optional utility called “**rte**” tags application binaries with resource requirements. This allows the scheduler to reserve resources that would be in excess of a single CPU core for a binary.
2. Linux hybrid execution services (binfmt_misc) registers the VERTEX loader called **VEX_load** to be ***transparently*** invoked for application binaries that have been “tagged”. Binaries for other non-native architectures may also be registered. For example, this would allow transparent execution of say ARM binaries on an x86 VERTEX node.
3. VEX_load queries the scheduler called **VEX_sched** to determine which remote node is available for execution with sufficient resources.
4. VEX_load calls the XCPU2 utility **xrx** to dispatch both the shadow process and the actual binary on the scheduled compute node. The shadow process exports the file namespace to the compute node over 9P.
5. So that the scheduler can run quickly without directly consulting hardware status, another deamon called **VEX_mond** monitors hardware and application processes and notifies the scheduler of changes such as failed processes and failed or recovered hardware resources.

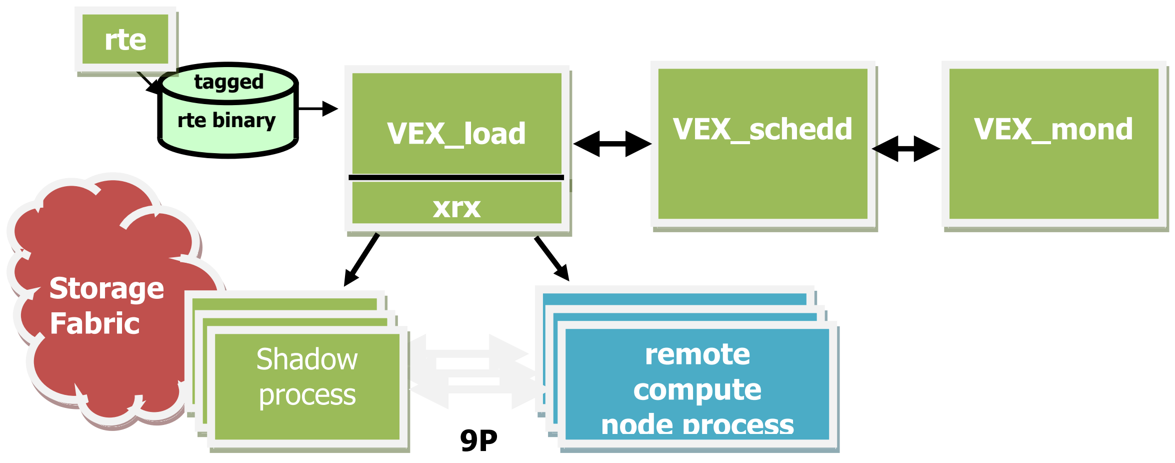

Figure 3. Software Architecture for VERTEX

As supercomputers mature and grow in size, the complexity of subsystems goes up leading to more complexity in the compute nodes. VERTEX reverses this trend by facilitating complexity in the VERTEX node and lowering complexity in the compute node. The use of the 9P protocol for both control and file system IO proxy requires only a single service on the compute node. The compute node does not require a workload manager, parallel file system manager, or system monitors. All of these services are handled by the VERTEX node. In fact, with virtual execution, it *appears* to the user that everything is running on the VERTEX node. However, what the user or system monitors see are shadow processes whose mission is to proxy file IO for the compute processes running on physically connected lightweight compute nodes. The 9P protocol is an effective technology that implements remote file system operations using a well developed protocol invented by Bell Labs for the Plan9 Operating System. This terse protocol can run on low-latency bare hardware protocols such as InfiniBand's OFED device drivers or higher level protocols such as TCP/IP. The scheduler just needs an identifier for available resources to inform the loader where to place work.

When a binary is started by the user or system workload manager, the VERTEX loader is automatically called. This loader must call a scheduler to determine where to start the work. The scheduler can schedule different types and quantities of work. The information about type of work and quantity is encoded in the binary. This is what we term "Resource Tagged Executables". Since different workload types can be used, VERTEX supports heterogeneous systems. VERTEX schedules heterogeneous resources quickly without physical communication to the remote resource. This allows for efficient utilization of hybrid multi-core system architectures that include the NVIDIA Fermi or the Intel MIC manycore system.

5. VERTEX Extensions and Innovations

The VERTEX architecture demonstrated at SuperComputing supports standard programming models such as MPI. We propose to extend the programming models on VERTEX with OpenMP by implementing a shared virtual memory model. The VERTEX architecture provides two key ingredients necessary for effective shared virtual memory: The first is a low-jitter low latency environment. The second is distributed control nodes not involved in the computation to help manage the virtual memory environment.

VERTEX is distinguished by several key enabling technologies. The first involves operating system support to improve scale. Traditional Linux environments do not provide low latency event-based communication [AM97] to applications, nor are they reliable in terms of providing low latency response. VERTEX will work with the open source community to provide a low latency, low jitter operating system interface that supports fast event-based communication among application threads.

The second involves effectively utilizing the increasingly multicore nodes available in these commodity platforms. We intend to extend VERTEX with a shared virtual memory (SVM) environment that will directly support the OpenMP programming interface and will effectively leverage the sharing across cores in a multicore environment. For efficiency, this SVM environment will leverage the low jitter event-based extensions to Linux described above. VERTEX's resource tagging will enable more effective resource control, allowing the runtime and operating system to allocate resources in a synergistic manner. The shared memory interface will provide a friendlier programming environment for end application developers. The result will be a scalable platform that will leverage the inherent capabilities of commodity platforms to their fullest.

VERTEX addresses the problem of performance scalability by providing an architecture with distributed control and IO nodes managing light weight compute nodes on a scalable network fabric. VERTEX also leverages programmer or user knowledge of problem size by creating executables with resource requirements, and uses resource-aware schedulers to quickly reserve resources and dispatch binaries. Lastly, VERTEX manages IO by exporting any file system name space with a single protocol to the compute nodes. In this way, all IO is proxied through the VERTEX node, sharing one protocol for both file system IO and execution control.

VERTEX is the first software platform for commodity hybrid multicore clusters that improves usability and efficiency by providing seamless virtual execution. Using shared virtual memory, the application or runtime can avoid the need to determine exactly what data to communicate, with whom the communication should occur, and when it should take place, thereby supporting the parallelization of larger classes of applications.

6. VERTEX vs. other Competitive Products and Research Projects

The distribution of services and filesystems grow as clusters grow in size and complexity. Different architectures use different schemes and names for these distributed service nodes. For example, the Hyperion system [HY09] uses gateway servers for distributing IO. XCAT clusters distribute system management on management nodes. Other than BlueGene , no other architecture has taken a systematic approach to distributing application control and IO through the same subsystem. Commodity platforms, which constitute the vast majority of the clusters in the market today, lack a software environment of this nature. VERTEX uses commodity components along with Linux on all (compute, control, and IO) nodes as compared to BlueGene's proprietary embedded compute nodes and proprietary compute node kernel.

7. VERTEX Partnership

The VERTEX software has been developed for architectures such as the SuperLinks platform from HPC Links. The SuperLinks platform uses hybrid blade systems hardware consisting of CPU's and GPU's, designed and manufactured by Boston Systems of UK and Supermicro. We have been using leading edge hybrid cluster HPC servers from Boston Systems in London for prototyping the alpha and beta versions of VERTEX. HPC Links did a demo of VERTEX on Boston machines in their booth at SC'10 in New Orleans. HPC Links did a press release on the SuperLinks platform in collaboration with Supermicro and NVIDIA at ISC'10 in Hamburg, and are doing joint sales and marketing of VERTEX based systems and solutions including SuperLinks, along with Supermicro and Boston. Current users of VERTEX software from HPC Links include an European Oil and Gas company, Univ. of Mass. Physics dept, and a DoD lab in the US.

8. References

[TMK94] P. Keleher, S. Dwarkadas, A. L. Cox, and W. Zwaenepoel,
`TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems", Winter USENIX Conference, January 1994.

[CSM97] R. Stets, S. Dwarkadas, N. Hardavellas, G. Hunt, L. Kontothanassis, S. Parthasarathy, and M. L. Scott, "Cashmere-2L: Software Coherent Shared Memory on a Clustered Remote-Write Network", 16th ACM Symposium on Operating Systems Principles, October 1997.

[AM97] S. Lumetta, A. Mainwaring, and D. Culler, Multi-Protocol Active Messages on a Cluster of SMPs, Supercomputing Conference (SC), November 1997.

[HU99] Y. C. Hu, H. Lu, A. L. Cox, and W. Zwaenepoel, OpenMP for Networks of SMPs, Proceedings of the 13th International Symposium on Parallel Processing and the 10th Symposium on Parallel and Distributed Processing, 1999.

[MPI] Open Source High Performance Computing, http://www.open-mpi.org

[OpenMP] The OpenMP Forum, http://www.openmp.org

[TOP500] Top 500 Supercomputing Sites, http://www.top500.org

[LK10] binfmt_misc, Linux Kernel Support for miscellaneous Binary Formats v1.1, Linux Kernel documentation, http://git.kernel.org/gitweb.cgi?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=Documentation/binfmt_misc.txt;hb=HEAD

[9P10] The 9P Protocol, http://plan9.bell-labs.com/magic/man2html/5/0intro

[XC10] XCPU Home Page, http://xcpu.org/

[GP02] Schmuck and Haskin, GPFS: A Shared-Disk File System for Large Computing Clusters, **P**roceedings of the 1st USENIX Conference on File and Storage Technologies, 2002, http://portal.acm.org/citation.cfm?id=1083349

[HY09] Hyperion System at Lawrence LiverMoore National Labs , https://str.llnl.gov/Dec09/seager.html

[BG06] Designing a Highly Scalable Operating System: The Blue Gene/L Story, Supercomputing 2006, http://sc06.supercomputing.org/schedule/pdf/pap178.pdf