Dynamic Visualisation Using Parallel Multicore System and HPC

Supriya Phatak[1], S. V. Patil[2], Rupali Jadhav[3], D. R. Anekar[4]
[1, 2, 3]Department of Computer Science & Enginnering
[4] Department of Information Technology
[1, 2, 3]ADCET Sangli, Shivaji University
[4] SAOE Pune, Pune University

Abstract- *As various obstacles showed up with implementing an application resource workflow, in the future the creation of Web Services for HPC services will be a solution for dedicated issues. For implementation testing the new supercomputer resources HLRN-II (North-German Supercomputing Alliance) have been used.*

Keywords— Multi Core parallel systems, Line of sight, HPC.

I. INTRODUCTION

Numerous applications and algorithms for handling dynamical visualisation and processing of scientific information could evolve more flexibility and facilities if they could use existing computing power more directly, namely MPP (Massively Parallel Processing) and SMP (Symmetric Multi-Processing) resources. Large benefits can result from using many cores of large computing resources in parallel, within a shorter time interval, for quasi interactive use. The idea of dynamical distributed resource usage for geoscientific information was introduced with the concept of Active Source [6]. For integration of HPC, Grid, and cluster resources are given as:

- Framework for the use of high end computing resources for dynamical visualization and information systems
- Inerrability of concepts (e.g. batch and scheduling)
- Frameworks for the application of algorithms needed
- Interfaces for flexible and secure data and application transfer, interchange, and distribution
- Portability of implementations, extendibility of existing methods, reusability of existing solutions.

Due to the limitations of delivering computing power from High Performance Computing, Grid Computing, and cluster computing resources interactively to a workstation, a framework is needed to integrate these resources. In absence of support for coupling these resources, in the past some features had to be last on the list to be addressed. HPC allows the scientific community to use a wide variety of computational resources to extend and accelerate the computation of their models. However, the selection of HPC architectures and programming interfaces for the models to be developed requires an important effort. Currently, HPC is characterized by the heterogeneity of their resources. Most of the modern supercomputers consist of clusters of multi-core nodes which include accelerator devices such as GPUs [4]. Current standalone computers present tremendous power, because of technological and architectural advances [5] and they are considered as desktop supercomputers if their heterogeneous resources are appropriately exploited. These architectures are based on multi-core processors and include additional resources to take advantage of different kinds of parallelism for each application (instructions level parallelism, data parallelism, task parallelism and so on).

II. HPC HARDWARE

Today most high-performance computer systems are built using the same components that power high-end commodity servers and workstations: i.e. multicore x86 CPUs from Intel & AMD, high-speed DD3 DRAM memory, high-end Graphics Processing Units (GPUs) from NVIDIA & AMD. There are several fundamental differences between single node high-performance servers (which can be considered high performance computing systems on their own) and clusters of these servers: the ability of processing elements to directly access ("load/store") shared memory banks on single-node systems and the presence of a dedicated high-speed network to increase the scale of the system on clusters. A cluster is a collection of (typically) high-end commodity servers, with a homogeneous configuration (same processor type & speed, same memory size), connected via a highspeed dedicated network. The reason why clusters are the HPC system of choice is that single-node systems can only be (economically) built to accommodate certain modest levels of processing elements and memory. Any needed increase in these elements has to be achieved by clustering multiple single-node systems together. Several vendors (Cray, IBM) build highly specialized clusters which use non-commodity parts particularly for their net works. These machines are called proprietary supercomputers or integrated massively parallel processing (MPP) systems. Many systems in the Top 500 [5] list of supercomputers in the world belong to this MPP category.

III. HPC SOFTWARE & MULTICORE MICROPROCESSOR ARCHITECTURE

Modern HPC software falls into two categories: a number of commercial packages that are used in specialized technical areas by industry (i.e. oil & gas, automobile design, drug design) and an assortment of custom applications written by a multitude of government research laboratories & academic institutions that range all the way from bullet-proof, robust codes to one-off research prototypes.

Microprocessor designers have long been considering many design choices to efficiently utilize the ever increasing effective silicon area with the increase of transistor density. The noticeable trend in this stream since mid-2000 is the multicore design. Instead of employing a complicated processor pipeline on a chip with an emphasis on improving single thread's performance, incorporating multiple processor cores on a single chip has become a main stream microprocessor design trend. As a Chip Multi-Processor (CMP), it can execute multiple software threads on a single chip at the same time. Thus a multicore processor provides a larger capacity of computations performed per chip for a given time interval (or throughput) [8].

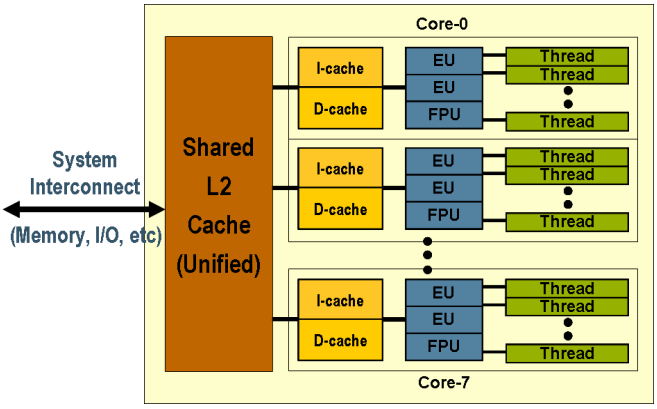

Fig1 Architecture of Advanced Multicore processor

In addition to the CMP based multi-core design, some designs go one step further to incorporate Simultaneous MultiThreading (SMT) or similar technologies such as Intel Hyperthreading on a processor core. Examples are Intel Nehalem and UltraSPARC T2/T3 microprocessor from Oracle/Sun. Fig.1 shows the architecture of an advanced multicore processor. On each processor chip, there are *N*processor cores, with each core having its own level-1 onchip cache. The *N*-cores share a larger capacity level-2 (and possibly level-3) cache(s) on (or off) the processor chip. Each core also has *M* hardware threads performing SMT or similar features. Thus it supports two levels of parallelism.

IV. STATUS OF THE IMPLEMENTATION

The status of implementation is given as below

A. HPC Resources and Configuration

For the work described here, the various resources of HLRN have been used. HLRN is the North-German Supercomputing Alliance. HLRN provides high-end High Performance Computing (HPC) resources jointly used and co-funded by the northern German states of Niedersachsen, Berlin, Bremen, Hamburg, Mecklenburg-Vorpommern, Schleswig-Holstein, and the Federal Government of Germany / German Research Society (DFG). Those resources include HLRN-II [3], a system comprised of two identical computing and storage complexes, one located at the Leibniz University¨at Hannover, Regionales Rechenzentrum f¨ur Niedersachsen (RRZN) and the other at the Konrad-Zuse-Zentrum f¨ur Informationstechnik Berlin (ZIB). By connecting the two systems via the HLRN-Link dedicated fibre optic network (Cisco Catalyst switches), HLRN can operate and administer them as one system. Each complex consists of MPP and SMP cluster components (SGI Altix ICE andXE) [4] installed in two phases. The first phase has been installed by Silicon Graphics Inc. in this year 2008.

For security reasons a trusted computing interface using sandboxing has been configured as various security policies for integrating data and applications have been introduced and successfully tested. This configuration allows very flexible transfer of data, secured execution of foreign Active Sources on demand, accounting as well as batch and interactive use of resources.

B. Batch System and Scheduling

The batch system, scheduling and resource management implemented on HLRN-II is based on Moab and Torque. With this system the PBS (Portable Batch System) resource specification language [5,10] is used. Interactive use and calculation is widely depending on features of the batch system used. Currently the end user application will have to do the job synchronisation. With a conventional system configuration the management of multi user operation is difficult. Both synchronising and multi user operation tend to work against interactive use.

C. Accessing Computing Resources

The Actmap Computing Resources Interface (CRI) is an actmap library containing procedures for handling computing resources. Examples for using High Performance

Computing and Grid Computing resources include batch system interfaces and job handling. This library (actlcri) can hold functions and procedures and even platform specific parts in a portable way. It can be used by calling the source code library as well as the byte code library generated with a compiler like TclPro. From an application, calling Actmap CRI can be done as follows. For various applications, byte code (TBC) [6] has been considered for any part of applications and data.

D. Distributing Data

Within event triggered jobs, MPI and batch means can be used for distributing and collecting data and job output. For distributing files automatically within the system e.g. dsh, pdsh, C3 tools, Secure Shell (SSH and SCP) are used. Interactive communication is supported by the appropriate Secure Shell key configuration. It must be part of the system configuration to correctly employ authorisation keys and crontab or at features.

E. Authorization and System Security

Authorisation for accessing data and information associated with the calculation currently affords to have one instance of the application present on one of the servers of the HPC resource, e.g. login or batch. A dedicated network using secure keys can be configured for the purpose of interactive application access in order to simplify communication and data transfer between the nodes. As for system security reasons large installations will tend to be restricted to dedicated users with this scenario. For execution of dynamic sources the trusted computing interface has been configured as policy trusted.

F. Accounting Jobs and Processes

The implemented framework is incorporated in an integrated solution for monitoring, accounting, billing supporting the geoinformation market. An outlook has been given for Geographic Grid Computing at the International Conference on Grid Services Engineering and Management (GSEM). Especially for the extended use of GIS and computing resources, the Grid-GIS framework, the "Grid-GIS house" has been created [9] and is used within the D-Grid [8,10] and with Condor. The Active Source components used here, are part of this framework, on top of the Grid services, Grid middleware, and the HPC and Grid resources.

V. PARALLELISM ANALYSIS METHOD

The parallelism analysis method is given as below:

A. Analysis Method for Data Dependence Violations

Definition 1. Produce-distance: the instruction numbers from the beginning of the thread to the last write operation for a specific memory address.

Definition 2. Consume-distance: the instruction numbers from the beginning of the thread to the first read operation for a specific memory address. The inter-thread data dependence can be abstracted as a producer/consumer model. Write operation is data producing while read operation is data consuming. To describe the data dependence violation, we introduce two terms here: "produce-distance" and "consume-distance". The produce-distance means the instruction numbers from the beginning of the thread to the last write instruction for a specific memory address, and consume-distance means the instruction numbers from the beginning of the thread to the first read instruction for a specific memory address. Either of them is calculated for specific data and both of them must be calculated at running time.

For thread i and its successor thread i+1, starting at almost the same time, if the latter's consume-distance is less than the former's produce-distance, there will be a dependence violation under the assumption of that all processor execute instructions at a same speed.

B. Benchmarks Choosing

In order to fully understand the TLP applicability, we analyzed three representative benchmark suites to give us a basically overall conclusion in the traditional application area. They are SPEC CPU2000 integer benchmark suite for desktop field, Mediabench benchmark suite for multimedia field, and SPLASH2 benchmark suite for HPC field. The reasons are as follows:

i. Desktop, media and HPC fields are the most important traditional application areas.
ii. These three benchmark suites all include the most important applications in their main field each.
iii. Their serial codes are compatible that is what the TLP technology needs.

C. Profiling Tool

The profiling tool we used in our investigation named ProLoop [5]. It's extended from sim-fast, the fasted simulator of SimpleScalar tool set which execute one instruction per

cycle. All experiments are done in Linux on the x86 platform. The destination ISA in the tool is PISA and the cross compiler from the gcc-2.7.2.3 with reconstructed backend is provided by Simple Scalar tool set.

VI. CHALLENGES IDENTIFIED

The most important challenges identified with these implementations on HPC resources have been grouped within this context in order to be briefly discussed.

- HPC resources and configuration
- Batch system and scheduling
- Accessing computing resources / Actmap Computing
- Resources Interface / Message Passing
- Distributing data
- Authorization and system security
- Accounting jobs and processes.

The following sections briefly describe the basic approaches for the implemented solution before showing an overall case study of an information system using distributed resources.

VII. LAUNCHING PARALLEL APPLICATIONS

The concepts presented in this paper generally apply to parallel applications in many kinds of distributed computing environments. However, for simplicity, this paper will present concepts in the context of a single use-case: launching a multiprocess parallel MPI job in a typical HPC cluster consisting of commodity multi-core NUMA machines. Launching multi-process, (possibly multi-threaded) parallel applications requires the support of a parallel run-time environment. Parallel run-time environments can launch and monitor groups of processes across nodes in an HPC system.

A. Mapping Processes

Among the first steps in launching a parallel job is obtaining computational resources on which to run. Modern HPC cluster resource managers can allocate compute resources at the granularity of individual processor cores, instead of only the more traditional node granularity. For example, the resource manager may (depending on site policy) allocate half the cores from node A and half the cores from node B to a single job.

B. Binding Processes

The process-launching agent of the parallel run-time environment works with the OS to limit exactly where each process can run in one of several ways: no restrictions, limited set restrictions, or specific resource restrictions.

i. No restrictions: The OS scheduler has full autonomy to decide where the process runs and what resources it uses.
ii. Limited set restrictions: The process-launching agent limits the job's individual processes to run on a common subset of processors on a node.
iii. Specific resource restrictions: The process-launching agent assigns specific, unique processors for each individual process.

VIII. CONCLUSION

As HPC systems continue grow more complex, a wider variety of process mapping and binding options will be needed to support these applications. Following best practices and utilizing expert experience can be an easy way to get started. By following these guidelines, computing power provided by the multi-core platform can be systematically exploited to improve application startup time, runtime throughput, and algorithm reliability when developing medical imaging applications.

ACKNOWLEDGEMENT

The author would like to thanks to Department of Computer Science and Engineering, Annasaheb Dange College of Engineering and Technology, Sangli.

REFERENCES

[1] Message Passing Interface Forum, MPI: A Message Passing Interface in Proceedings of Supercomputing '93, IEEE Computer Society Press, November 1993, pp. 878–883.

[2] S. Ethier, W. M. Tang, R. Walkup, and L. Oliker, Large-scale Gyrokinetic Particle Simulation of Microturbulence in Magnetically Confined Fusion Plasmas, IBM Journal of Research and Development, Vol. 52, January 2008, pp. 105–115.

[3] E. Jeannot and G. Mercier, Near-optimal Placement of MPI Processes on Hierarchical NUMA Architectures, in PROCEEDINGS of the 16th International Euro-Par Conference on Parallel Processing, ser. Euro-Par'10. Berlin, Heidelberg: Springer-Verlag, pp. 199–210, 2010.

[4] E. Gabriel, G. E. Fagg, G. Bosilca, Open MPI: Goals, Concept, and Design of a Next Generation MPI

Implementation, Proceedings of the 11th European PVM/MPI

[5] F.Alvarado, Parallel Solution of Transient Problems by Trapezoidal Integration Power Apparatus and Systems, IEEE Transactions on, Vol.PAS-98, no. 3, May 1979, pp. 1080 –1090.

[6] A. Bose, ParaAllel Processing in Dynamic Simulation of Power Systems Sadhana,Vol.18,1993, pp. 815–841.

[7] X. Wang, S. Ziavars, C. Nwankpa, J. Johnson, and P. Nagvajara, Parallel Solution of Newton´s Power Flow Equations on Configurable Chips, International Journal of Electrical Power & Energy Systems, Vol. 29, No. 5, pp. 422 – 431, 2007.

[8] D. Falcao, Lecture Notes. Springer-Verlag, 1997, ch. High performance computing in power system applications.

[9] http://www.top500.org

[10] http://www.infinibandta.org