DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Job Scheduling for Adaptive Applications in Future HPC Systems

Nishanth Nagendra

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Job Scheduling for Adaptive Applications in Future HPC Systems

Job Scheduling für Adaptive Anwendungen auf Zukünftigen HPC Systemen

Author:	Nishanth Nagendra
Supervisor:	Prof. Dr. Michael Gerndt
Advisor:	M.Sc. Isaias Alberto Compres Urena
Submission Date:	May 15, 2015

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, May 15, 2015 Nishanth Nagendra

Acknowledgments

Abstract

Invasive computing is a novel paradigm for the design and resource-aware programming of future parallel computing systems. It enables the programmer to write resource aware programs and the goal is to optimize the program for the available resources. Traditionally, parallel applications implemented using MPI are executed with a fixed number of MPI processes before submitting to a HPC(High Performance Computing) system. This results in a fixed allocation of resources for the job. Newer techniques in scientific computing such as AMR(Adaptive Mesh Refinement) result in applications exhibiting complex behavior where their resource requirements change during execution. Invasive MPI which is a part of an ongoing research effort to provide MPI extensions for the development of Invasive MPI applications will result in evolving jobs for the HPC systems during runtime that utilize such AMR techniques. Unfortunately, using only static allocations result in the evolving applications being forced to execute using their maximum resource requirements that may lead to an inefficient resource utilisation. In order to support such parallel evolving applications at HPC centers there is an urgent need to investigate and implement extensions to existing resource management systems or develop an entirely new one. This thesis will extend the work done over the last few months during which an early prototype was implemented by developing a protocol for the intgeration of invasive resource management into existing standard batch systems. Specifically, This thesis will now investigate and implement a job scheduling algorithm in accordance with the new protocol developed earlier for supporting such an invasive resource management.

Contents

1 Introduction

Over the last two decades, the landscape of Computer Architecture has changed radically from sequential to parallel . Due to the limiting factors of technology we have moved from single core processors to multi core processors having a network interconnecting them. Traditionally, the approach of designing algorithms has been sequential, but designing algorithms in parallel is gaining more importance now to better utilize the computing power available at our disposal. Another important trend that has changed the face of computing is an enormous increase in the capabilities of the networks that connect computers with regards to speed, reliability etc. These trends make it feasible to develop applications that use physically distributed resources as if they were part of the same computer. A typical application of this sort may utilize processors on multiple remote computers, access a selection of remote databases, perform rendering on one or more graphics computers, and provide real-time output and control on a workstation. Computing on networked computers ("Distributed Computing") is not just a subfield of parallel computing as the basic task of developing programs that can run on many computers at once is a parallel computing problem. In this respect, the previously distinct worlds of parallel and distributed computing are converging.

As technology advances, we have newer problems or applications that demand larger computing capabilities which push the limits of technology giving rise to newer advancements. The performance of a computer depends directly on the time required to perform a basic operation and the number of these basic operations that can be performed concurrently. A metric used to quantify the performance of a computer is FLOPS (floating point operations per second). The time to perform a basic operation is ultimately limited by the "clock cycle" of the processor, that is, the time required to perform the most primitive operation. The term ***High Performance Computing (HPC)*** refers to the practice of aggregating computing power (multiple nodes with processing units interconnected by a network in a certain toplogy) or the use of parallel processing for running advanced application programs efficiently, reliably and quickly. The term applies especially to systems that function above a ***teraflop*** or 10^{12} floating-point operations per second. The term HPC is occasionally used as a synonym for Supercomputer that work at more than a ***petaflop*** or 10^{15} floating-point operations per second. The most common users of HPC systems are scientific researchers, engineers, government agencies including the military, and academic institutions. In general, HPC systems can refer to Clusters, Supercomputers, Grid Computing etc. and they are usually used for running complex applications.

A ***Batch System*** is used to manage the resources in a HPC System. It is a middleware that comprises of two major components namely the ***Resource Manager*** and ***Scheduler***. The role of a Resource Manager is to act like a glue for a parallel computer to execute parallel jobs. It should make a parallel computer as easy to use as a PC. A programming model such as ***Message Passing Interface (MPI)*** for programming on distributed memory systems would typically be used to manage communications within a parallel program by using the MPI library functions. A Resource Manager allocates resources within a HPC system, launches and otherwise manages Jobs. Some of the examples of widely used open source as well as commercial resource managers are ***SLURM, TORQUE, OMEGA, IBM Platform LSF*** etc. Together with a scheduler it is termed as a Batch System. The role of a job scheduler is to manage queue(s) of work when there is more work than resources. It supports complex scheduling algorithms which are optimized for network topology, energy efficiency, fair share scheduling, advanced reservations, preemption, gang scheduling (time-slicing jobs) etc. It also supports resource limits (by queue, user, group, etc.). Many batch systems provide both resource management and job scheduling within a single product (e.g. LSF) while others use distinct products(e.g. Torque Resource Manager and Moab Job Scheduler). Some other examples of Job Scheduling Systems are ***LoadLeveler, OAR, Maui, SLURM etc.***

Existing Batch Systems usually support only static allocation of resources to Jobs/Applications before they start which means the resources once allocated are fixed for the lifetime of the application. The complexity of applications have been growing especially when we consider advanced techniques in Scientific Computing like ***Adaptive Mesh Refinement (AMR)*** where applications exhibit complex behavior by changing their resource requirements during execution. The Batch Systems of today are not equipped to deal with such kind of complex applications in an intelligent manner apart from giving the Job the maximum number of resources before it starts that will result in sheer wastage of resources leading to poor resource utilisation. In order to support such parallel adaptive applications at HPC centers there is an urgent need to investigate and implement extensions to existing resource management systems or develop an entirely new one. These supporting infrastructures must be able to handle the new kind of adaptive applications and the legacy static jobs intelligently keeping in mind that they should now be able to achieve much higher system utilisation, energy efficiency etc. compared to their predecessors due to the elasticity of the applications.

1.1 Invasive Computing

The throughput of HPC Systems depends not only on efficient job scheduling but also on the type of jobs forming the workload. As defined by Feitelson, and Rudolph, Jobs can be classified into four categories based on their flexibility:

- ***Rigid Job:*** Requires a fixed number of resources throughout its execution.
- ***Moldable Job:*** The resource requirement of the job can be molded or modified by the Batch System before starting the job(e.g. to effectively fit alongside other rigid jobs). Once started its resource set cannot be changed anymore.
- ***Evolving Job:*** These kind of jobs request for resource expansion or shrinkage during their execution. Applications that use Multi-Scale Analysis or Adaptive Mesh Refinement (AMR) exhibit this kind of behavior typically due to unexpected increases in computations or having reached hardware limits (e.g. memory) on a node.
- ***Malleable Job:*** The expansion and shrinkage of resources are initiated by the batch system in contrast to the evolving jobs. The application adapts itself to the changing resource set.

The first two types fall into the category of what is called as the static allocation since the allocation of rigid and moldable jobs must be finalized before the job starts. Whereas, the last two types fall under the category of dynamic allocation since this property of expanding or shrinking evolving and malleable jobs (together termed adaptive jobs) happens at runtime. Adaptive Jobs hold a strong potential to obtain high system performance. Batch systems can substantially improve the system utilization, throughput and response times with efficient shrink/expand strategies. Similarly, applications also profit when expanded with additional resources as this can increase application speedup and improve load balance across the job's resource set.

Invasive Computing is a novel paradigm for the design and resource-aware programming of future parallel computing systems. It enables the programmer to write efficient resource aware programs. This approach can be used to allocate, execute on and free resources during execution of the program. This results in adaptive applications at runtime that can expand and shrink in the number of their resources. This will also potentially help to achieve much better resource utilisation and energy efficiency if efficient shrink/expand strategies can be used for all the jobs/applications. HPC infrastructure like Clusters, Supercomputers execute a vast variety of jobs, majority of which are parallel applications. These centers use intelligent resource management systems that should not only perform tasks of job management, resource management and scheduling but also satisfy important metrics like higher system utilization, job throughput and responsiveness. Traditionally, MPI applications are executed with a fixed number of MPI processes but with Invasive MPI they can evolve dynamically at runtime in the number of their MPI processes. This in turn supports advanced techniques like AMR where the working set size of applications change at runtime. These advancements result in adaptive applications at runtime and therefore need an intelligent resource management system that can provide efficient resource utilization at HPC centers. They should also now be able to achieve much higher system utilisation, energy efficiency etc. compared to their predecessors due to elasticity/adaptivity of the applications.

Under the collaborative research project funded by the **German Research Foundation (DFG)** in the **Transregional Collaborative Research Centre 89 (TRR89)**, research efforts are being made to investigate Invasive Computing approach vertically at different levels of abstraction right from the hardware up to the programming model and including the applications. **Invasive MPI** is an effort towards invasive programming with MPI where the application programmer has MPI extensions available for specifying at certain safe points in the program to allow for elasticity which means that the application can evolve.

1.2 Dynamic Resource Management

Two of the most widely used resource managers on HPC systems are **SLURM** and **TORQUE**. The two major components in general of any sophisticated resource manager are the batch scheduler and the process manager. The Process Manager is responsible for launching the jobs on the allocated resources and managing them throughout their lifetime. Examples of process manager are ***Hydra, SLURM Daemon (slurmd)*** etc. The Process Managers interact with the processes of a parallel application via the **Process Management Interface (PMI)**. In order to support Invasive Resource Management, The following components will be implemented: ***iScheduler*** (Batch Scheduler for Invasic Jobs) built as an extension into an existing batch system and ***iDRScheduler*** (Invasive Distributed Run TIme Scheduler) similar to a controller daemon which will sit between the batch scheduler and the process manager. **SLURM** is the choice of an existing batch system on which this prototype will be implemented for demonstrating Invasive Computing and below is a high level illustration of the architecture for such an Invasive Resource Management.

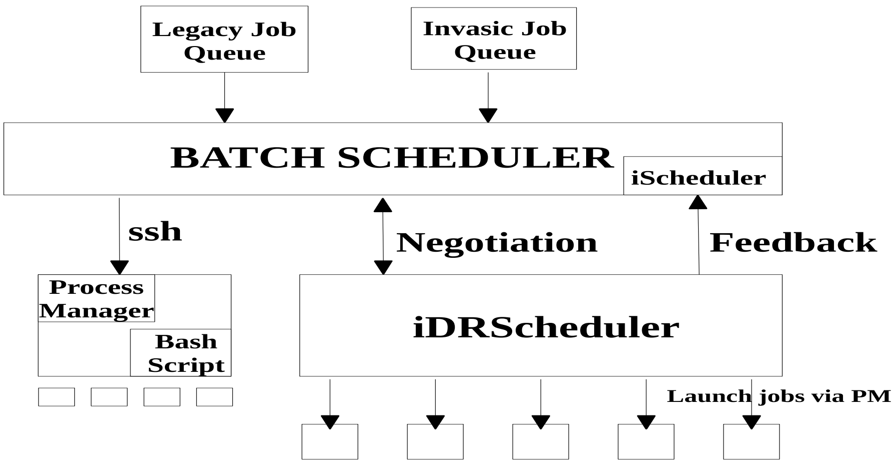

Figure 1.1: Invasive Resource Management Architecture

The figure1.1 illustrates the proposed Invasive Resource Management Architecture. In addition to a Job Queue for Legacy static jobs, we now have an additional Job Queue for Invasic (Adaptive) Jobs. The existing Batch Scheduler needs to be extended in order to schedule these new type of Invasic Jobs and a new component called ***Invasic Scheduler (iScheduler)*** is responsible for this. In contrast to modifying an existing system to support Invasive Resource Management, a new component called ***Invasive Distributed Run Time Scheduler (iDRScheduler)*** is proposed which sits between the Batch Scheduler and Process Manager. The objective of such a multi-level approach is to avoid modifying the existing system which will be a substantially large effort and rather have an independent component that caters specifically to Invasic Jobs. It is responsible for managing the resources present in the Invasic paritition used specifically for running Invasic Jobs. With this approach, the existing Legacy Jobs can be served via the existing batch scheduler and the new Invasic Jobs can be served by via iScheduler and iDRScheduler. iDRScheduler talks to the iScheduler via a protocol called the Negotiation protocol to receive Invasic Jobs dispatched from iScheduler which it then launches for execution by performing some run time scheduling like pinning of jobs, expand/shrink etc.

The new components proposed in the architecture help achieve the objective of supporting dynamic resource management for invasic applications. iScheduler is responsible for scheduling Invasic Jobs. The scheduling decisions are communicated via negotiation protocol to iDRScheduler and these decisions are basically job(s) selected via a scheduling algorithm to be submitted for execution. The decisions will be made on the basis of

Resource Offers sent by iDRScheduler which creates these resource offers based on the state of the partition. Upon receiving a resource offer, iScheduler will either accept it by selecting jobs from the queue that can be mapped to this offer or reject it. A Resource Offer can represent real or virtual resources because the iDRScheduler can also present a virtual view of resources in the hope of getting a mapping of jobs to offer that is more suitable to satisfy its local metrics such as resource utilization, power stability, energy efficiency etc. It can either accept or reject the mapping received from iScheduler. Similar to iDRScheduler, the iScheduler makes its decisions to optimize for certain local metrics such as high job throughput, reduced job waiting times, deadlines, priorities etc. This highlights the mismatching policies/metrics for which both the iDRScheduler and iScheduler make their decisions on and hence both will be involved in some kind of a negotiation via the protocol to reach a common agreement. iDRScheduler is an independent entity introduced with the purpose of inter-operating with existing batch systems rather than replacing them with an entirely new one. It may be possible that in the future this component will not be a separate entitiy but will be built into the batch system itself.

Shrink/Expand Strategies: There are several strategies one may employ to tackle adaptive applications during runtime and take decisions that can lead to higher system utilisation, energy efficiency etc.:

- Let us consider that at any given point of time there are some Invasic Jobs running in the system. If the parallel runtime environment is able to anticipate that in the near future, there may be a large window of free resources available because some applications may shrink according to a prediction of their scalability behavior with the help of run time profiling and collected performance data then iDRScheduler can provide a resource offer to iScheduler. This offer can specify a virtual list of nodes more than what is available in order to get a mapping of jobs. These jobs can then be shrunk and fit into the existing space of resources with the knowledge that they may be able to expand later.
- Another scenario is where there is an anticipation of a smaller window of resources in the future because some of the currently running applications may expand. In such a case iDRScheduler will provide a resource offer to iScheduler with a virtual list of nodes smaller than what is currently available in order to get a mapping of jobs. These jobs can be then be expanded and fit into the existing space of resources with the knowledge that they may have to shrink later.
- The third variation could be the case where the runtime anticipates the state of resources to remain the same as the current state for the near future since

none of the running applications may expand or shrink. In such a scenario iDRScheduler will send a resource offer that is a list of nodes which is exactly the same as the available resources to iScheduler in order to get a mapping of jobs. These jobs can then be fit into the space of resources available as it is without expanding/shrinking them.

1.3 Master Thesis

The focus of this Master Thesis is to extend the early prototype developed as a part of the guided research in the last few months. It will now give concrete meanings to the negotiation protocol, defining the format of the invasic job records, its constraints, definition of resource offers, feedback reports and most important of all to investigate and develop an efficient job scheduling algorithm at the iScheduler level. Following this closely from the Guided Research would be to continue having an automated testing in place that will help in simulating a workload of jobs, testing the job scheduling algorithm for its correctness, evaluating and analysing the performance of such a prototype for various metrics. Given below is a tentative plan of the activities proposed monthwise starting from November 15 till May 15 to be carried out for the duration of 6 months in this Master Thesis.

1.3.1 Tentative Plan

- Literature survey for current/recent related work on batch job scheduling from research groups focusing on problems in the areas of batch scheduling, resource management, middleware etc. In addition to this, defining resource offers, invasic job records and job constraints that can come from a pre-computed performance model of an application during its runtime or could be user defined. Once these tasks have been completed, it will follow up with extending the early prototype by the implementation of these ideas and testing them for correctness using the automated testing feature.

- Integration of the fake iHypervisor with the real iHypervisor by making it as its plugin. Test this integration for all the earlier development using the automated testing feature for correctness. Review, analyse, fix any errors and repeate the process till the integration is stable. Follow this up by understanding the SLURM's existing scheduling algorithms including the recent Machine Learning approach for a possible reuse or enhancement. Define the format of feedback reports sent by iHypervisor and how they can be processed on the iScheduler side including its storage mechanism (possibly using SLURM's existing database functionalities). Implement these ideas in the prototype and test it thoroughly.

- Investigate and experiment with basic scheduling algorithms first to later proceed with complex ones. Design and implement a sophisticated scheduling algorithm for mapping jobs from Invasic job queue to the resource offers sent by iHypervisor. This algorithm must perform its decision making by also using the collected history/feedback data about many statistical measures provided by iHypervisor periodically.

- Implementation of the Job Scheduling Algorithm continues for realizing all the necessary requirements as proposed earlier till it has been successfully implemented and later followed by a lot of functional testing.

- Test the full prototype along with the other component iHypervisor (Including its Run Time scheduling and Invasive Resource Management) for simple to complex workloads (emulating a queue of invasic jobs possibly including legacy static jobs). Testing also needs to be done with real Invasic applications developed by the *Chair Of Scientific Computing*. Find issues/errors, review, analyse, correct and repeat the process till it is stable and then collect all the results necessary.

- Coming up with the draft version of the Master Thesis Report followed by reviews, corrections. This process is repeated till the final version is decided. Also prepare the slides for the Master thesis to present them at a later stage.

The above timeline highlights a tentative plan for the activities to be taken up during the Master Thesis and the 6 items above correspond to these 6 months of the Thesis in a chronological order. The steps may overlap or shift depending on the progress but the same overall structure will be followed for the Thesis. It will also include in parallel small amounts of documentation in the report as and when necessary during the 6 month period and not necessarily everything at the end.

2 Related Work

3 Invasive Computing

3.1 Job Classification

3.2 Resource Aware Programming

3.3 Traditional Resource Management

3.4 Support for Invasive Computing

3.4.1 Invasive MPI

3.4.2 Resource Management Extensions

4 Dynamic Resource Management Architecture

4.1 System Design

4.1.1 Batch Scheduler

4.1.2 Distributed Run Time Scheduler

4.1.3 MPI Process Manager

4.2 Negotiation Protocol

4.3 Invasive Jobs

5 iScheduler Design

5.1 Job Description

5.2 Resource Offers

5.3 Feedback Reports

5.4 Negotiation Protocol

5.5 Job Scheduling Algorithm

5.5.1 Problem Formulation

5.5.2 Pseudo Code

6 Implementation

6.1 Plugins for SLURM

6.2 Data Structures

6.3 State Machine Diagrams

6.3.1 iScheduler

6.3.2 iHypervisor

6.4 Important APIs

7 Evaluation

7.1 Method of Evaluation

7.1.1 Emulation of Workload

7.1.2 Real Invasic Applications

7.2 Setup

7.3 Experiments and Results

7.4 Performance and Graphs

8 Conclusion and Future Work

8.1 Future Work

[Pra+14] [Pra+15] [Ioa+11] [DL96] [Ure+12] [Ger+12] [Cer+10] [Mag+08] [UCA04] [KP01] [Bal+10] [YJG03] [Gup+14] [Lif95] [Sko+96] [Hun04] [Cao+10] [Zho+13] [Luc11] [TLD13] [Zho+15] [Tan+10] [Tan+05] [FW98] [FW12] [Sch]

Glossary

computer is a machine that. . . .

Acronyms

TUM Technische Universität München.

List of Figures

List of Tables

Bibliography

[Bal+10] P. Balaji, D. Buntinas, D. Goodwell, W. Gropp, J. Krishna, E. Lusk, and R. Thakur. “PMI: A Scalable Parallel Process-Management Interface for Extreme-Scale Systems.” In: *Recent Advances in the Message Passing Interface* (Sept. 2010).

[Cao+10] Y. Cao, H. Sun, W.-J. Hsu, and D. Qian. “Malleable-Lab: A Tool for Evaluating Adaptive Online Schedulers on Malleable Jobs.” In: *Euromicro Conference on Parallel, Distributed and Network-Based Processing(PDP)* (Feb. 2010).

[Cer+10] M. C. Cera, Y. Georgiou, O. Richard, N. Maillard, and P. Navaux. “Supporting malleability in parallel architectures with dynamic cpusets mapping and dynamic MPI.” In: *International Conference on Distributed Computing and Networking(ICDCN)* (Jan. 2010).

[DL96] D.G.Feitelson and L.Rudolph. “Towards convergence in job schedulers for parallel supercomputers.” In: *Workshop on Job Scheduling Strategies for Parallel Processing* (Apr. 1996).

[FW12] D. G. Feitelson and A. M. Weil. “Scheduling Batch and Heterogeneous Jobs with Runtime Elasticity in a Parallel Processing Environment.” In: *International Parallel and Distributed Processing Symposium Workshops and Phd Forum* (May 2012).

[FW98] D. G. Feitelson and A. M. Weil. “Utilization and Predictability in Scheduling the IBM SP2 with Backfilling.” In: *Parallel Processing Symposium and Symposium on Parallel and Distributed Processing(IPPS/SPDP)* (Apr. 1998).

[Ger+12] M. Gerndt, A. Hollmann, M. Meyer, M. Schrieber, and J. Weidendorfer. “Invasive Computing with iOMP.” In: *Forum on Specification and Design Languages(FDL)* (Feb. 2012).

[Gup+14] A. Gupta, B. Acun, O. Sarood, and L. V. Kale. “Towards Realizing the Potential of Malleable Jobs.” In: *High Performance Computing(HiPC)* (Dec. 2014).

[Hun04] J. Hungershofer. “On the Combined Scheduling of Malleable and Rigid Jobs.” In: *International Symposium on Computer Architecture and High Performance Computing(SBAC-PAD)* (Oct. 2004).

[Ioa+11] N. Ioannou, M. Kauschke, M. Gries, and M. Cintra. "Phase-Based Application-Driven Hierarchical Power Management on the Single-chip cloud Compupter." In: *Parallel Architectures and Compilation Techniques(PACT)* (Oct. 2011).

[KP01] C. Klein and C. Perez. "An RMS for Non-predictably Evolving Applications." In: *Cluster Computing(CLUSTER)* (Sept. 2001).

[Lif95] D. A. Lifka. "The ANL/IBM SP scheduling system." In: *Job Scheduling Strategies for Parallel Processing* (Apr. 1995).

[Luc11] A. Lucero. "Simulation of Batch Scheduling using Real Production Ready Software Tools." In: *Iberian Grid Infrastructure Conference* (June 2011).

[Mag+08] K. E. Maghraoui, T. J. Desell, B. K. Szymanski, and C. A. Varela. "Dynamic Malleability in Iterative MPI Applications." In: *Concurrency and Computation: Practice and Experience* (Jan. 2008).

[Pra+14] S. Prabhakaran, M. Iqbal, S. Rinke, C. Windisch, and F. Wolf. "A Batch System with Fair Scheduling for Evolving Applications." In: *International Conference on Parallel Processing(ICPP)* (Sept. 2014).

[Pra+15] S. Prabhakaran, M. Neumann, S. Rinke, F. Wolf, A. Gupta, and L. V. Kale. "A Batch Sytem with Efficient Adaptive Scheduling for Malleable and Evolving Applications." In: *International Parallel and Distributed Processing Symposium(IPDPS)* (May 2015).

[Sch] SchedMD. *SLURM Workload Manager*. www.slurm.schedmd.com.

[Sko+96] J. Skovira, W. Chan, H. Zhou, and D. Lifka. "The EASY - LoadLeveler API Project." In: *Job Scheduling Strategies for Parallel Processing* (Apr. 1996).

[Tan+05] W. Tang, N. Desai, D. Buettner, and Z. Wan. "Backfilling with Lookahead to Optimize the Packing of Parallel Jobs." In: *Journal of Parallel and Distributed Computing* (May 2005).

[Tan+10] W. Tang, N. Desai, D. Buettner, and Z. Wan. "Analyzing and Adjusting User Runtime Estimates to Improve Job Scheduling on the Blue Gene/P." In: *International Symposium on Parallel and Distributed Processing Symposium(IPDPS)* (Apr. 2010).

[TLD13] W. Tang, Z. Lan, and N. Desai. "Job Scheduling with Adjusted Runtime Estimates on Production Supercomputers." In: *Journal of Parallel and Distributed Computing* (Mar. 2013).

[UCA04] G. Utrera, J. Corbalan, and J. Albarta. "Implementing Malleability on MPI Jobs." In: *International Conference on Parallel Architecture and Compilation Techniques(PACT)* (Oct. 2004).

[Ure+12] I. A. C. Urena, M. Riepen, M. Konow, and M. Gerndt. "Invasive MPI on Intel's single-chip cloud computer." In: *Architecture of Computing Systems(ARCS)* (Feb. 2012).

[YJG03] A. B. Yoo, M. A. Jette, and M. Gondona. "SLURM: Simple Linux Utility for Resource Management." In: *Job Scheduling Strategies for Parallel Processing* (June 2003).

[Zho+13] Z. Zhou, Z. Lan, W. Tang, and N. Desai. "Reducing Energy Costs for IBM Blue Gene/P via Power-Aware Job Scheduling." In: *Job Scheduling Strategies for Parallel Processing* (May 2013).

[Zho+15] Z. Zhou, X. Yang, D. Zhao, P. Rich, W. Tang, J. Wang, and Z. Wan. "I/O Aware Batch Scheduling for Petascale Computing Systems." In: *International Conference on Cluster Computing* (Sept. 2015).