# HTCaaS: A Large-Scale High-Throughput Computing by Leveraging Grids, Supercomputers and Cloud

Seungwoo Rho, Seoyoung Kim, Sangwan Kim, Seokkyoo Kim, Jik-Soo Kim and Soonwook Hwang
Korea Institute of Science and Technology Information (KISTI), Daejeon, Republic of Korea
Email: {seungwoo0926, sssyyy77, sangwan, anemone, jiksoo.kim, hwang}@kisti.re.kr

***Abstract*—We present the *HTCaaS* (High-Throughput Computing as a Service) system which aims to provide researchers with ease of exploring large-scale and complex HTC problems by leveraging Supercomputers, Grids, and Cloud. HTCaaS can hide heterogeneity and complexity of harnessing different types of computing infrastructures from users, and efficiently submit a large number of jobs at once by effectively managing and exploiting of all available computing resources.**

**Our system has been effectively integrated with national Supercomputers in Korea, international computational Grids, and Amazon EC2 resulting in combining a vast amount of computing resources to support most challenging scientific problems.**

## I. Introduction

High-Throughput Computing (HTC) [1] consists of running many loosely-coupled tasks that are independent (there is no communication needed between them) but requires a large amount of computing power during relatively a long period of time. Middleware systems such as Condor [2] or BOINC [3] have successfully achieved a tremendous computing power by harnessing a large number of computing resources. However, as the number of jobs and the complexity of scientific applications increase, it becomes a challenge for the traditional middleware systems employing typically a single type of resources (e.g., clusters of workstations, desktop machines over Internet) to solve the given scientific problem within a reasonable amount of time. Also, recent emerging applications requiring millions or even billions of tasks to be processed with relatively short per task execution times have led the traditional HTC to expand into *Many-Task Computing* (MTC) [4].

Therefore, to effectively support complex and demanding scientific applications, it is inevitable to harness as many computing resources as possible including Supercomputers, Grids, and even Cloud. However, it is challenging for researchers to effectively utilize available resources that are under control by independent resource providers as the number of jobs (that should be submitted *at once*) increase dramatically (as in parameter sweeps or N-body calculations).

We designed and implemented the *HTCaaS* (High-Throughput Computing as a Service) system that can hide heterogeneity and complexity of leveraging different computing resources from users, and efficiently submit a large number of jobs at once by effectively managing and exploiting of all available computing resources.

The contribution of our work is as followings:

- *Ease of Use*: We minimize user overhead for handling a large amount of jobs & computing resources
- *Intelligent Resource Selection*: HTCaaS can automatically select more responsive and effective resources and adapt to the current load by dynamically adjusting acquired resources
- *Pluggable Interface to Resources*: We adopt GANGA [5]s plugin mechanism for accessing heterogeneous computing resources without hardcoding
- *Support for Many Client Interfaces*: A wide range of client interfaces are supported including a native WS-interface, Java API, and Client tools (CLI, GUI)

HTCaaS system has been successfully integrated with PLSI [6] Supercomputers in Korea, international computational Grids, and Amazon EC2 [7] resulting in harnessing a vast amount of computing resources to support most challenging scientific problems.

## II. Design and Implementation of the HTCaaS System

Figure 1 shows the overall architecture of the HTCaaS system and it consists of five server-side modules (*Account Manager*, *User Data Manager*, *Job Manager*, *Agent Manager*, *Monitoring Manager*) and two client-side tools (Command-Line Interface and Graphic User Interface).

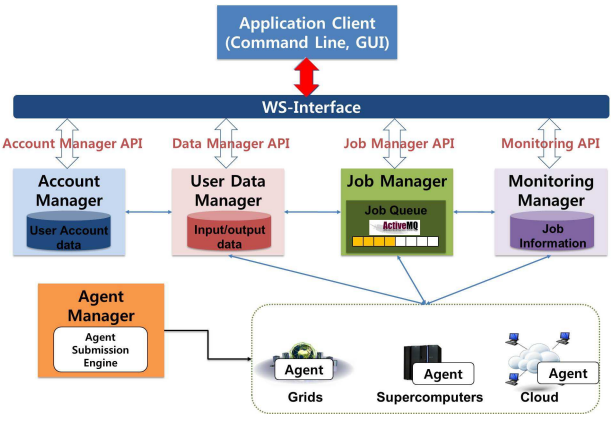


Fig. 1. HTCaaS System Architecture

A job in our system is the data and associated profile that describes a computation to be performed. Since users may want to submit a large number of jobs by employing parameter sweeps or N-body calculations, HTCaaS introduces a concept of the *Meta-Job* which specifies a higher-level job description based on the OGF JSDL standard [8]. Once a Meta-Job is submitted, HTCaaS automatically splits it into many jobs and inserts them into the Job Queue (implemented in ActiveMQ [9]) managed by the Job Manager. All of required input data and produced results are stored at the User Data Manager.

Once jobs are submitted into our system, *agents* (implemented in Java) are dispatched from Agent Manager and process jobs in Supercomputers, Grids, and Clouds. HTCaaS employs agent-based *multi-level scheduling* & *streamlined job dispatching* (as described in Falkon [10]) so that a first-level request to a batch scheduler (e.g., Load Leveler [11] in PLSI Supercomputers, gLite [12] for Grids, PBS [13] for Amazon EC2) reserves resources by submitting agents as batch jobs and then each agent *proactively* pulls the tasks from the Job Manager which implements the lightweight and fast job dispatching mechanisms.

We improve further upon previous multi-level scheduling systems such as Falkon [10] or MyCluster [14] by:

- Providing powerful Meta-job description which allows users to easily specify a large amount of computations (e.g., parameter sweeps)
- Automatic job split and submission of many tasks into the Job Queue which will be proactively dispatched by active agents
- User-level job scheduling & accounting which can dynamically adapt to the current system load and consider future incoming jobs
- Loosely-coupled service-oriented architecture based on WS-Interface which support diverse clients through a native Web Service APIs, Java APIs

Therefore, users of HTCaaS are able to submit and execute hundreds of thousands of jobs (which can be simply expressed by a single JSDL script) within an automated process, effectively monitor them and process the final results. For those who are not familiar with XML style of scripting, we also provide an easy-to-use GUI tool which can automatically generate JSDL script based on user's input so that it can be submitted into our system.

The overall steps of job submission and execution in HTCaaS system (as we can see from Figure 2) are as followings:

1) User logins HTCaaS and uploads input data through User Data Manager.
2) User submits a Meta-Job (written in JSDL) which can be composed of multiple tasks.
3) HTCaaS automatically divides a Meta-Job into multiple tasks based on the specification and insert them into the Job Queue.
4) Agent Manager dispatches agents based on job requirements and resource availability.

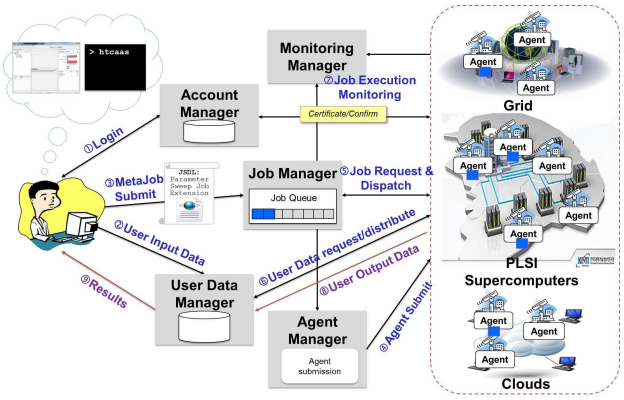


Fig. 2. Job Submission & Execution Steps in HTCaaS

5) Agents proactively request tasks and process them. While processing the tasks, if needed, agents contact the User Data Manager to request the input data.
6) Finished results are stored in the User Data Manager and notified to the user.

HTCaaS system has been applied to various scientific applications including Pharmaceutical domain (new drug discovery by simulating protein docking), Computer-Aided Design (3D visualization of optimized design solution) and High-Energy Physics area (Quantum chromodynamics (QCD): strong interactions of fundamental particles at hadron colliders). All of these applications usually have a large number of computations (a target protein of 3CL-pro of SARS with 1.1 Million chemical compounds for protein docking and 5 million particle collisions resulting in tens of millions of interactions for QCD simulations) or require substantial amount of computing time (on average 500 CPU utilized, completed in 2.8 days, totally 2.6 years of computation for the CAD application) but they are independent so that as we increase the number of resources that can process the tasks, we can achieve higher throughput and performance.

This is a typical scenario in a high-throughput computing environment, enabling many independent users to submit their jobs to a collection of node resources in the system, or *embarrassingly parallel* (sometimes called *Bags of Tasks*) workloads. Indeed, Iosup et al. [15], [16] found that a high percent of Grid applications still employ an embarrassingly parallel model based on their analysis on the characteristics of traces of real Grid environments.

HTCaaS is still under an active development and it has been tested on top of heterogeneous computing environments. Currently, it is seamlessly integrated with PLSI and we are launching a pilot service of HTCaaS for the PLSI users which can enable wider usage of national supercomputing infrastructures in Korea. In the near future, we will address system scalability, fault tolerance and improved user-level job scheduling of our HTCaaS system to ultimately provide an efficient and effective middleware system that can process large-scale high-throughput computing over a vast amount of hybrid computing infrastructures.

## REFERENCES


[1] M. Livny, J. Basney, R. Raman, and T. Tannenbaum, "Mechanisms for High Throughput Computing," *SPEEDUP Journal*, vol. 11, no. 1, 1997.

[2] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[3] D. Anderson, "BOINC: A System for Public-Resource Computing and Storage," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID 2004)*, Nov. 2004.

[4] I. Raicu, I. Foster, and Y. Zhao, "Many-Task Computing for Grids and Supercomputers," in *Proceedings of the Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS'08)*, Nov. 2008.

[5] J. Moscicki, F. Brochu, J. Ebke, U. Egede, J. Elmsheuser, K. Harrison, R. Jones, H. Lee, D. Liko, A. Maier, A. Muraru, G. Patrick, K. Pajchel, W. Reece, B. Samset, M. Slater, A. Soroko, C. Tan, D. van der Ster, and M. Williams, "Ganga: A tool for computational-task management and easy access to Grid resources," *Computer Physics Communications*, vol. 180, no. 11, pp. 2303–2316, 2009.

[6] Partnership & Leadership for the nationwide Supercomputing Infrastructure, "Available at http://www.plsi.or.kr/."

[7] Amazon Elastic Compute Cloud, "Available at http://aws.amazon.com/ec2/."

[8] Open Grid Forum Job Submission Description Language, "Available at http://www.gridforum.org/documents/GFD.56.pdf."

[9] Apache ActiveMQ: the most popular and powerful open source messaging and Integration Patterns server, "Available at http://activemq.apache.org/."

[10] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde, "Falkon: a Fast and Light-weight tasK executiON framework," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing (SC'07)*, Nov. 2007.

[11] IBM Tivoli Workload Scheduler LoadLeveler, "Available at http://www-03.ibm.com/systems/software/loadleveler/."

[12] gLite - Lightweight Middleware for Grid Computing, "Available at http://glite.cern.ch/."

[13] B. Bode, D. M. Halstead, R. Kendall, Z. Lei, and D. Jackson, "The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters," in *Proceedings of the Usenix, Proceedings of the 4th Annual Linux Showcase & Conference*, Nov. 2000.

[14] E. Walker, J. P. Gardner, V. Litvin, and E. L. Turner, "Creating Personal Adaptive Clusters for Managing Scientific Jobs in a Distributed Computing Environment," in *Proceedings of the Challenges of Large Applications in Distributed Environments (CLADE'06)*, Jun. 2006.

[15] A. Iosup, C. Dumitrescu, and D. Epema, "How are Real Grids Used? The Analysis of Four Grid Traces and Its Implications," in *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing (GRID 2006)*, Sep. 2006.

[16] A. Iosup and D. Epema, "Grid Computing Workloads: Bags of Tasks, Workflows, Pilots, and Others," *IEEE Internet Computing*, vol. 15, no. 2, 2011.