Falkon: A Proposal for Project Globus Incubation

1. A proposed name for the project

Falkon: a Fast and Light-weight tasK executiON framework

2. The prefix to use for the email lists that will be set up(*-dev, *-user, etc)

falkon-dev, falkon-user, etc

3. A proposed project chair, with contact information

Ioan Raicu iraicu@cs.uchicago.edu

4. A list of the proposed committers for the project

Ioan Raicu[1] and Yong Zhao[1] and Catalin L. Dumitrescu[1] and Ian Foster[1,2] and Michael Wilde[1,2]

[1] Computer Science Dept., University of Chicago 1100 E. 58th Street, Ryerson Hall Chicago, IL 60637, USA ***{iraicu,yongzh,cldumitr,foster,wilde}@cs.uchicago.edu***
[2] Mathematics and Computer Science Department, Argonne National Laboratory Cass Ave, Chicago, IL 61637, USA ***{foster,wilde}@mcs.anl.gov***

5. An Overview of the Aims of the Project

Many interesting computations can be expressed conveniently as data-driven task graphs, in which individual tasks wait for input to be available, perform computation, and produce output. Systems such as DAGMan, Karajan [7, 8], Swift [7, 8], and VDS support this model. These systems have all been used to encode and execute thousands of individual tasks. However, dispatching such tasks directly to batch schedulers has two disadvantages. First, because a typical batch scheduler provides rich functionality (e.g., multiple queues, flexible task dispatch policies, accounting, per-task resource limits), the time required to dispatch a task can be large—30 secs or more—and the aggregate throughput relatively low (perhaps two tasks/sec). Second, while batch schedulers may support different queues and policies, the policies implemented in a particular instantiation may not be optimized for many tasks.

To enable the rapid execution of many tasks on compute clusters, we have developed Falkon, a Fast and Light-weight tasK executiON framework. Falkon uses (1) multi-level scheduling to separate resource acquisition (via, e.g., requests to batch schedulers) from task dispatch, and (2) a streamlined dispatcher. Multi-level scheduling, introduced in operating systems research in the 1990s, has been applied to clusters by the Condor team and others, while streamlined dispatchers are found in, e.g, BOINC. Falkon's integration of the techniques delivers performance not provided by any other system [1].

5.1. Current Status

We have already architected and implemented a first prototype that is deployed on the TeraGRID. Micro-benchmarks show that Falkon throughput (487 tasks/sec) and scalability (to 54,000 executors and 2,000,000 tasks processed in just under two hours) are one to two orders of magnitude better than other schedulers. Applications (executed by the Swift parallel programming system) reduce end-to-end run time of up to 90% for large-scale astronomy and medical applications, relative to versions that execute tasks via separate scheduler submissions.

Falkon's web site is at http://people.cs.uchicago.edu/~iraicu/research/Falkon/, and the latest stable code release (v0.8.1) is at http://people.cs.uchicago.edu/~iraicu/research/Falkon/Falkon_v0.8.1.tgz. We also have the project in SVN source control and can be downloaded with "svn co https://svn.ci.uchicago.edu/svn/vdl2/falkon".

5.2. Future Work and Directions

We plan to implement and evaluate enhancements, such as task prefetching, alternative technologies, data management, and a three-tier architecture [2].

Technologies: Performance depends critically on the behavior of our task dispatch mechanisms; the number of

messages needed to interact between the various components of the system; and the hardware, programming language, and compiler used. We implemented Falkon in Java and use the Sun JDK 1.4.2 to compile and run Falkon. We use the GT4 Java WS-Core to handle Web Services communications. One potential optimization is to rewrite Falkon in C/C++, (using, for example, the Globus Toolkit C WS-Core). Another is to change internal communications between components to a custom TCP-based protocol. However, dispatch rates are adequate for applications studied to date, and the primary obstacle to scaling is likely to be data access, not task dispatch at the current achieved task dispatch rates. We plan to support other communication protocols by December 2007, and will be included in v1.0 release.

Data management: Many Swift [7, 8] applications read and write large amounts of data. Applications typically access data from a shared data repository (e.g., NFS, GPFS, GridFTP, web server). Thus, data access can become a bottleneck as applications scale. We expect that data caching, proactive data replication, and data aware scheduling can offer significant performance improvements for applications that have locality in their data access patterns. We plan to implement data caching mechanisms in Falkon executors, which would allow executors to populate local caches with data the corresponding task would require. In conjunction with data caching we may wish to implement a data-aware dispatcher. We will evaluate to what extent data aware dispatching reduces raw dispatch throughput rates. A user can choose which dispatcher and executor to use for a specific application. These features are planned to be released in v0.9 by November 2007.

3-Tier Architecture: Falkon currently requires that the dispatcher and client can each send messages to the other. Thus, each must have at least one port open in their firewall. We have implemented a polling mechanism to bypass firewalls on executors or clients, but we loose this performance and scalability port open in the firewall on which it will accept WS due to the polling mechanism vs. the notification mechanisms. Note that the dispatcher is still required to receive messages from clients and executors. Falkon also currently assumes that executors operate in a public IP space, so that the dispatcher can communicate with them directly. If (as is sometimes the case) a cluster is configured with a private IP space, to which only a head node has access, the Falkon dispatcher must run on that head node. If we decide to implement this new architecture, it will likely be released in v1.0 in December 2007.

Documentation: We plan to draft up more documentation and tutorials on configuring, using, and debugging Falkon for end users and developers to find it easier to use and integrate into their applications. We will plan to complete this by December 2007 for inclusion in v1.0.

6. An Overview of Any Current User Base or User Community

- Swift Team (University of Chicago / USA): around 10 researchers from University of Chicago, Argonne National Labs, and Computational Institute, http://www.ci.uchicago.edu/swift/
- ServMark Team (Globus Incubator Project): more than 15 people from more than 4 universities, http://dev.globus.org/wiki/Incubator/ServMark
- TeraGRID: partially deployed, http://www.teragrid.org/
- AstroPortal: Astronomy application to analyze astronomy image datasets in collaboration with NASA, John Hopkins University, and University of Chicago, http://people.cs.uchicago.edu/~iraicu/research/AstroPortal/ [4, 5, 6, 9]; this project was the first prototype implementation of Falkon (v0.1).

6.1. Short Term Envisaged Enlargement for the User Base

We also envisage that the following user communities will have references implementations of Falkon to be used in the respective environments.

Workspace service: The workspace service provides a gateway to a set of resources configured with the Xen implementation of virtual machines [10]. Our plan is to inter-operate with them to do provisioning of not just Falkon executors over LRMs, but provisioning of VMs through the Workspace service (University of Chicago / Argonne National Laboratory, http://workspace.globus.org/). More information on this project can be found at http://www.ci.uchicago.edu/wiki/bin/view/VDS/DslCS/Falkon_EC2.

IBM BlueGene/P: Porting of various components of Falkon to interoperate with the BlueGene/P software stack (http://www.llnl.gov/asc/computing_resources/bluegenel/). More information on this project can be found at http://www.ci.uchicago.edu/wiki/bin/view/VDS/DslCS/Falkon_BG.

7. An overview of how the Candidate relates to other parts of Globus

First, Falkon is implemented as a WSRF service and a fast scheduling interface for large bags of jobs in a Grid.

Second, users of Grid workflow engines, such as Swift, Karajan, BPEL or Condor-G, will have now the choice to reduce system overloads when submitting large amounts at once of jobs into a Grid [3]. When using Swift and Falkon together, we demonstrated reductions in end-to-end run time by as much as 90% for applications from the astronomy and medical fields, when compared to the same applications run over batch schedulers [2].

Third, FaLkon represents a complement for the work of Bresnahan et al. that targets a multi-level scheduling architecture specialized for the dynamic allocation of compute cluster bandwidth. A modified Globus GridFTP server varies the number of GridFTP data movers as server load changes [11].

Fourth, virtual workspaces [10] will benefit substantially from our work, by enabling Swift based applications to execute on virtual resources managed by the Workspace Service.

8. A summary of Why the Candidate would Enhance and Benefit Globus

The schedulers used to manage parallel computing clusters are not typically configured to enable easy configuration of application-specific scheduling policies. In addition, their sophisticated scheduling algorithms and feature-rich code base can result in significant overhead when executing many short tasks. Falkon is designed to enable the efficient dispatch and execution of many small tasks, as required in Grid environments and for Grid workflow engines. To this end, it uses a multi-level scheduling strategy to enable separate treatment of resource allocation (via conventional schedulers) and task dispatch (via a streamlined, minimal-functionality dispatcher).

9. References

[1] Ioan Raicu, Catalin L. Dumitrescu, Ian Foster, *Dynamic Resource Provisioning in Grid Environments*, TeraGRID Conference 2007, Madison, WI, USA (poster).

[2] Ioan Raicu, Yong Zhao, Catalin L. Dumitrescu, Ian Foster, Michael Wilde, *Falkon: a Fast and Light-weight tasK executiON framework*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (Super-Computing) 2007, Reno, NV, USA (accepted for publication).

[3] Catalin Dumitrescu, Ioan Raicu, Ian Foster, *Usage SLA based Scheduling in Grids*, Journal of Concurrency and Computation: Practice and Experience, Special Issue (GCC'05), 2006.

[4] Ioan Raicu, Ian Foster, Alex Szalay. "Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets", in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (Super-Computing), 2006, Tampa, FL, USA (poster).

[5] Ioan Raicu, Ian Foster, Alex Szalay, Gabriela Turcu. "AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis", TeraGrid Conference 2006, Indianapolis, USA.

[6] Alex Szalay, Julian Bunn, Jim Gray, Ian Foster, Ioan Raicu. "The Importance of Data Locality in Distributed Computing Applications", NSF Workflow Workshop 2006.

[7] Yong Zhao, Mihael Hategan, Ben Clifford, Ian Foster, Gregor von Laszewski, Ioan Raicu, Tiberiu Stef-Praun, Mike Wilde. "Swift: Fast, Reliable, Loosely Coupled Parallel Computation", in Proceedings of the IEEE Workshop on Scientific Workflows 2007 (to appear).

[8] Yong Zhao, Mihael Hategan, Ioan Raicu, Mike Wilde, Ian Foster. "Swift: a Parallel Programming Tool for Large Scale Scientific Computations", under review at Scientific Programming Journal, Special Issue on Dynamic Computational Workflows: Discovery, Optimization, and Scheduling.

[9] Ioan Raicu, Ian Foster. "Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets", NASA GSRP Proposal, Ames Research Center, NASA, February 2006 -- Award funded 10/1/06 – 9/30/07.

[10] Virtual Workspaces, http://workspace.globus.org/

[11] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster, The Globus Striped GridFTP Framework and Server, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (Super-Computing) 2005, Seattle, WA, 2005