

# Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets

## Year 2 Progress Report & Year 3 Proposal

### 1 Year 1 and Year 2 Proposals

In order to setup the context for this progress report and new proposal, this section covers a brief motivation for our work and summarizes the Year 1 and Year 2 Proposals we originally submitted under grant number NNA06CB89H.

#### 1.1 Year 1

Large datasets are being produced at a very fast pace in the astronomy domain. In principle, these datasets are most valuable if and only if they are made available to the entire community, which may have tens to thousands of members. The astronomy community will generally want to perform various analyses on these datasets to be able to extract new science and knowledge that will both justify the investment in the original acquisition of the datasets as well as provide a building block for other scientists and communities to build upon to further the general quest for knowledge.

Grid Computing has emerged as an important new field focusing on large-scale resource sharing and high-performance orientation. The Globus Toolkit, the “de facto standard” in Grid Computing, offers us much of the needed middleware infrastructure that is required to realize large scale distributed systems. We proposed to develop a collection of Web Services-based systems that use grid computing to federate large computing and storage resources for dynamic analysis of large datasets. We proposed to build a Globus Toolkit 4 based prototype named the “AstroPortal” that would support the “stacking” analysis on the Sloan Digital Sky Survey (SDSS). The stacking analysis is the summing of multiple regions of the sky, a function that can help both identify variable sources and detect faint objects. We proposed to deploy the AstroPortal on the TeraGrid distributed infrastructure and apply the stacking function to the SDSS DR5 dataset, which comprises more than 320 million objects dispersed over 1.5 million files, a total of 9 terabytes of data.

We claimed that our work with the AstroPortal would lead to interesting and innovative research work in three main areas: 1. *resource management* (efficient task dispatch, dynamic resource provisioning); 2. *data management* (data diffusion, data-aware scheduling); and 3. *applications* (performance and scalability). Our achievements for the first year are outlined in the Final Report for Year 1.

#### 1.2 Year 2

As a continuation to the initial proposal, we proposed to generalize our work from the AstroPortal even further beyond the implementation of the basic components, namely Falkon. Although these basic building blocks should allow the implementation of many applications to be built with relatively little effort, as we had shown in our Year 1 Final Report, we believe it would be valuable to define an abstract model that formally defines each basic component and its interaction with other components. This abstract model should allow us to explore the general problem space much more freely as we will break free of any application specific implementation or feature which might have influenced us when we implemented Falkon and the AstroPortal.

The analysis of large datasets typically follows a split/merge pattern, which includes an analysis query to be answered, which get split down into independent tasks to be computed, after which the results from all the tasks are merged back into a single aggregated result. Based on the split/merge methodology, we propose AMDASK, an Abstract Model for DAta-centric taSK farms, which defines the abstract model that allows us to study the stated hypothesis. Traditionally, task farms have been defined as a common parallel pattern which drives the computation of independent tasks, where a task is a self contained computation. The data-centric component of the abstract model emphasizes the central role data plays in the task farm model we are proposing, and the fact that the task farm is optimized to take advantage of data cache storage and data locality found in many large datasets and typical application workloads. Together, a data-centric task farm is defined as a common parallel pattern which drives the independent computational tasks taking into consideration the data locality in order to optimize the performance of the analysis of large datasets.

We intend to validate the AMDASK model more generally through simulations. We will implement the AMDASK model in a discrete event simulation that will allow us to investigate a wider parameter space than we could in a

real world implementation and deployment. We expect the simulations to help us prove that the AMDASK model is both efficient and scalable given a wide range of simulation parameters. The outputs from the simulations over the entire considered parameter space will form the datasets that will be used to statistically validate the model using  $R^2$  statistic and graphical residual analysis.

## 2 Year 2 Progress Report

The results of the Year 1 proposal can be found in the Year 1 Final Report. This section will only discuss the progress we have made on the Year 2 proposal. This section will first discuss the completed milestones, followed by the following short-term goals, the deliverables we expect to produce, and the dissemination of our results.

### 2.1 Completed Milestones

Our proposal which built upon the work on the AstroPortal and Falkon centered on two main areas: the 1) *definition of an abstract model for data-centric task farms*, and the 2) *validation of the abstract model*. Our progress has been mostly in the formally defining the abstract model. A complete definition of the abstract model can be found in “Harnessing Grid Resources with Data-Centric Task Farms”.

**Base Definitions:** A data-centric task farm has various components (i.e. computational resource where the tasks are to execute, storage resources where the data needed by the tasks is stored, etc). We formally defined 12 basic elements that are later used to derive relations regarding the model: 1) Persistent data stores, 2) Transient data stores, 3) Transient resources, 4) Data Objects, 5) Store Capacity, 6) Compute Speed, 7) Load, 8) Ideal Bandwidth, 9) Available Bandwidth, 10) Copy Time, 11) Tasks, and 12) Computational Resource State.

**Execution Model:** We also defined an execution model, to tie the relationships between the basic definitions defined prior. The execution model outlines the respective policies that control various parts of the execution model and how they relate to the definitions in the previous section. Each incoming task is dispatched to a transient resource, selected according to the *dispatch policy*. If a response is not received after a time determined by the *replay policy*, or a failed response is received, the task is re-dispatched according to the *dispatch policy*. A missing data object that is required by a task and does not exist on the transient data store is copied from transient or persistent data stores selected according to the *data fetch policy*. If necessary, existing data at a transient data store are discarded to make room for the new data, according to the *cache eviction policy*. Each computation is performed on the data objects found in a transient data store. When all computations are complete, the result is aggregated and returned; this aggregation of the results is assumed to be free to simplify the abstract model. Finally, we define a *resource acquisition policy* that decides when, how many, and for how long to acquire new transient computational and storage resources for. Similarly, we also define a *resource release policy* that decides when to release some acquired resources.

**The Performance and Efficiency of the Abstract Model:** We investigate when we can achieve good performance with this abstract model for data-centric task farms and under what assumptions. We define various costs (costs per task and average task execution time) and efficiency related metrics (efficiency, computational intensity, efficiency overheads). Furthermore, we explore the relationships between the different parameters in order to optimize efficiency.

**Cost per task:** We define the *cost per task*  $\chi(\kappa)$  as follows: 
$$\chi(\kappa) = \begin{cases} o(\kappa) + \mu(\kappa), & \delta \in \phi(\tau) \\ o(\kappa) + \mu(\kappa) + \zeta(\delta, \tau), & \delta \notin \phi(\tau) \end{cases}$$

**Average Task Execution Time:** We define the *average task execution time*,  $B$ , as the summation of all the task execution times divided by the number of tasks; more formally, we have  $B = \frac{1}{|K|} \sum_{k \in K} \mu(\kappa)$ .

**Computational Intensity:** Let  $A$  denote the *arrival rate of tasks*; we define the *computational intensity*,  $I$ , as follows:  $I = B * A$ . If  $I = 1$ , then all nodes are fully utilized; if  $I > 1$ , tasks are arriving faster than they can be executed; finally, if  $I < 1$ , then there are nodes that might be idle.

**Workload Execution Time:** We define the *workload execution time*,  $V$ , of our system as 
$$V = \max\left(\frac{B}{|T|}, \frac{1}{A}\right) * |K|.$$

**Workload Execution Time with Overhead:** In general, the total execution time for a task  $\kappa \in K$  includes overheads, which reduced efficiency by a factor of  $\frac{\mu(\kappa)}{\chi(\kappa)}$ . We define the *workload execution time with overhead*,

$W$ , of our system as  $W = \max\left(\frac{Y}{|T|}, \frac{1}{A}\right) * |K|$ , where  $Y$  is the *average task execution time including overheads*

$$\text{defined as } Y = \begin{cases} \frac{1}{|K|} \sum_{\kappa \in K} [\mu(\kappa) + o(\kappa)], & \delta \in \phi(\tau), \delta \in \Omega \\ \frac{1}{|K|} \sum_{\kappa \in K} [\mu(\kappa) + o(\kappa) + \zeta(\delta, \tau)], & \delta \notin \phi(\tau), \delta \in \Omega \end{cases}.$$

**Efficiency:** We define the *efficiency*,  $E$ , of a particular workload as  $E = \frac{V}{W}$ . The expanded version of efficiency is

$$E = \frac{\max\left(\frac{B}{|T|}, \frac{1}{A}\right) * |K|}{\max\left(\frac{Y}{|T|}, \frac{1}{A}\right) * |K|}, \text{ which can be reduced to } E = \begin{cases} 1, & \frac{Y}{|T|} \leq \frac{1}{A} \\ \max\left(\frac{B}{Y}, \frac{|T|}{A * Y}\right), & \frac{Y}{|T|} > \frac{1}{A} \end{cases}.$$

We claim that for the caching mechanisms to be effective in this model (i.e. the needed data objects to be found in transient data stores), the *aggregate capacity of our transient storage resources T is greater than our workload's working set,  $\Omega$* , (all data objects required by a sequence of tasks) *size*; formally, we can say  $\sum_{\tau \in T} \sigma(\tau) \geq |\Omega|$ .

We also claim that we can obtain  $E > 0.5$  if  $\mu(\kappa) > o(\kappa) + \zeta(\delta, \tau)$ , where  $\mu(\kappa)$ ,  $o(\kappa)$ ,  $\zeta(\delta, \tau)$  are the time to execute and dispatch the task  $\kappa \in K$ , and copy the object  $\delta$  to  $\tau \in T$ , respectively.

**Speedup:** We define the *speedup*,  $S$ , of a particular workload as  $S = E * |T|$ .

**Optimizing Efficiency:** Having defined both efficiency and speedup, it is possible to maximize for either one, as efficiency normally monotonically decreases and speedup increases with more resources used. We can *optimize efficiency* by finding the smallest number of *transient compute/storage resources*  $|T|$  while we maximize speedup times efficiency.

## 2.2 Short Term Goals

Our goals for the second half of the Year 2 center around the validation of the abstract model we have defined. More details can be found in "Harnessing Grid Resources with Data-Centric Task Farms".

We will validate the AMDASK model through simulations, to verify the both the efficient and scalability given a wide range of simulation parameters (i.e. number of storage and computational resources, communication costs, management overhead, and workloads – including inter-arrival rates, query complexity, and data locality). We will implement the model in a discrete event simulation that will allow us to investigate a wider parameter space than we could in a real world implementation.

The simulations will specifically attempt to model a grid computing environment comprising of computational resources, storage resources, batch schedulers, various communication technologies, various types of applications, and workload models. We will perform careful and extensive empirical performance evaluations in order to create correct and accurate input models to the simulator; the input models include 1) Communication costs, 2) Data management costs, 3) Task scheduling costs, 4) Storage access costs, and 5) Workload models.

We expect to be able to scale simulations to more computational and storage resources than we could achieve in a real deployed system due to the availability of resources. Furthermore, assuming the input models to be correct, we should be able to accurately measure the end-to-end performance of various applications using a wide range of strategies for the various resource management components.

## 2.3 Deliverables and Dissemination of the Results

We expect to have the following deliverables upon the completion (09/30/2008) of the NASA GSRP Fellowship under Grant Number NNA06CB89H. As a result of the work funded by the NASA GSRP Fellowship, we have produced a series of software systems, papers, documents, presentations, an online portal and documentation, which

can all be accesible online at my main research page at <http://people.cs.uchicago.edu/~iraicu>. Deliverables that have not been completed yet will also be posted on this web site when they are completed. We clearly denote each item below if it is not completed yet with the following names: 1) IN PROGRESS, or 2) TO DO.

- Abstract Model:
  - Definition
  - Validation via Simulations: (IN PROGRESS)
- Papers and Reports.
  - I. Raicu, Y. Zhao, I. Foster, A. Szalay. "Accelerating Large Scale Scientific Exploration through Data Diffusion," under review at IEEE Workshop on Data-Aware Distributed Computing 2008.
  - I. Raicu, Y. Zhao, I. Foster, A. Szalay. "A Data Diffusion Approach to Large Scale Scientific Exploration," Microsoft eScience Workshop at RENCi 2007.
  - I. Raicu. "Harnessing Grid Resources with Data-Centric Task Farms", University of Chicago, Computer Science Department, PhD Proposal, December 2007, Chicago, Illinois.
  - I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde. "Falkon: a Fast and Light-weight tasK executiON framework", IEEE/ACM SuperComputing 2007.
  - I. Raicu, C. Dumitrescu, I. Foster. "Dynamic Resource Provisioning in Grid Environments", TeraGrid Conference 2007.
- Documents for NASA:
  - Year 2 Proposal
  - Year 2 Progress Report
  - Year 2 Final Report: (TO DO)
- Presentations:
  - "Harnessing Grid Resources with Grid Resources with Data Data-Centric Task Farms Centric Task Farms", University of Chicago, Department of Computer Science, Dissertation Proposal, December 12th, 2007.
  - "Falkon: a Fast and Light-weight tasK executiON framework for Grid Environments", IEEE/ACM SuperComputing 2007, November 15th, 2007.
  - "Accelerating Large Scale Scientific Exploration with Falkon", IEEE/ACM SuperComputing 2007, Argonne National Laboratory Booth, November 14th, 2007.
  - "A Data Diffusion Approach to Large Scale Scientific Exploration", University of Chicago, CS Department, DSL Seminar, October 24th, 2007.
  - "A Data Diffusion Approach to Large Scale Scientific Exploration", 2007 Microsoft eScience Workshop at RENCi, October 21st, 2007.

### 3 Year 3 Proposal

As a continuation to our initial proposal, we would like to generalize our work further to allow a large class of applications to transparently use the mechanisms that allowed the AstroPortal to perform and scale so well. Those mechanisms have been implemented in Falkon, to support efficient task dispatch, dynamic resource provisioning, and data management through data diffusion. We plan on exploring the performance of data diffusion with more applications and workloads through the synergy we have created between Falkon and the Swift parallel programming system. We have integrated Falkon into the Karajan workflow engine, which in term is used by the Swift parallel programming system. Thus, Karajan and Swift applications can use Falkon without modification. We have already observe reductions in end-to-end run time by as much as 90% when compared to traditional approaches in which applications used batch schedulers directly by performing dynamic resource provisioning and providing applications with a lighter weight task dispatch mechanism. Swift has been applied to applications in the physical sciences, biological sciences, social sciences, humanities, computer science, and science education. We have successfully executed several applications (medical imaging, astronomy image analysis, molecular dynamics simulations) through Swift over Falkon (without data diffusion). We plan on investigating the performance benefits of data diffusion on these applications as well as others from bio-informatics, pharmaceuticals, and physics for our Year 3 Proposal. There is considerable work that needs to be done to interface the Swift system's data management

capabilities to those of Falkon's data management capabilities in order for Swift applications to take advantage of the data diffusion from Falkon.

We also plan to evolve the Falkon architecture from the current 2-Tier architecture to a 3-Tier one. We are expecting that this architecture change would allow us to introduce more parallelism and distribution of the currently centralized management component in Falkon, and hence offer higher dispatch and execution rates than Falkon currently supports. We are pursuing this work with the goal to have Falkon run at considerably larger scales, such as those found on the latest IBM BlueGene/P (BG/P) that will be online in 2008 at Argonne National Laboratory. The work in porting Falkon to the BG/P will open new opportunities to applications that traditionally could not execute on the BG/P due to the lack of support of task farms. It will be crucial to test the limits of the 3-Tier architecture from a performance point of view to evaluate the appropriateness of Falkon on the BG/P which can scale to 10s of millions of processors (the current configuration will boast 128K CPU cores). Furthermore, we will also be working at simplifying the various components in Falkon, including the communication protocols that are internal to the system. We plan to implement the Executor in C (in addition to the one that is already implemented in Java), and offer a proprietary TCP-based communication protocol (as opposed to the existing Web Services protocol) between the Executors and the Dispatcher. This transition should allow Falkon to achieve higher performance due to the lighter weight communication protocol, and allow the Executor to be deployed on computer architectures that do not support Java, such as the IBM BlueGene.

#### **4 Contributions & Conclusions**

We see the dynamic analysis of large datasets to be important due to the ever growing datasets that need to be accessed by larger and larger communities. Attempting to address the storage and computational problems separately (essentially forcing much data movement between computational and storage resources) will not scale to tomorrow's peta-scale datasets and will likely yield significant underutilization of the raw computational resources.

It has been argued that data intensive applications cannot be executed in grid environments because of the high costs of data movement. But if data analysis workloads have internal locality of reference, then it can be feasible to acquire and use even remote resources, as high initial data movement costs can be offset by many subsequent data analysis operations performed on that data. We envision "data diffusion" as a process in which data is stochastically moving around in the system, and that different applications can reach a dynamic equilibrium this way. One can think of a thermodynamic analogy of an optimizing strategy, in terms of energy required to move data around ("potential wells") and a "temperature" representing random external perturbations ("job submissions") and system failures. Our work proposes exactly such a stochastic optimizer.

In the Year 1 Proposal, we proposed to build a system called AstroPortal, to enable the efficient analysis of large astronomy datasets on Grid resources; the AstroPortal was generalized in another system called Falkon. In the Year 2 Proposal, we defined an abstract model for data-centric task farms, AMDASK, in order to address the integration of the storage and computational issues found in a class of applications which can be decomposed down into many independent computational tasks which need to work on large datasets (i.e. AstroPortal). We plan to validate the abstract model via discrete event simulations, and to provide a reference implementation to show the flexibility and effectiveness of it on real world applications and datasets. Our current proposal for Year 3 proposes that we expand the applications (and user base) that can benefit from the proposed AMDASK model and its practical realization Falkon by creating a synergy with the Swift parallel programming system. Furthermore, we expect to increase the user base of Falkon by ensuring that Falkon works efficiently on the new IBM BlueGene/P supercomputer that will be housed at Argonne National Laboratory.

There are various fundamental research questions we have addressed and hope to address through our work presented in this proposal and previous ones. They center on two main areas, data and compute resource management, and how they relate to particular workloads of data analysis on large datasets. We have explored several applications from various domains, such as astronomy, astro-physics, medicine, chemistry, and economics in order to show off the flexibility and effectiveness of the AMDASK model and its implementation (Falkon) on real world applications; much of the outreach in the wide range of scientific domains has been accomplished through the fantastic synergy that has been created between Falkon and the Swift parallel programming system. Finally, we have several more applications (i.e. bio-informatics, pharmaceuticals, physics) that we are considering investigating and are already working with the domain scientists to gather requirements for these new applications.