Many-Task Computing for Grids and Supercomputers

Ioan Raicu[1], Ian T. Foster[1,2,3], Yong Zhao[4]

[1]*Department of Computer Science, University of Chicago, Chicago IL, USA*
[2]*Computation Institute, University of Chicago, Chicago IL, USA*
[3]*Mathematics and Computer Science Division, Argonne National Laboratory, Argonne IL, USA*
[4]*Microsoft Corporation, Redmond, WA, USA*

iraicu@cs.uchicago.edu, foster@anl.gov, yozha@microsoft.com

Abstract

Many-task computing aims to bridge the gap between two computing paradigms, high throughput computing and high performance computing. Many task computing differs from high throughput computing in the emphasis of using large number of computing resources over short periods of time to accomplish many computational tasks (i.e. including both dependent and independent tasks), where primary metrics are measured in seconds (e.g. FLOPS, tasks/sec, MB/s I/O rates), as opposed to operations (e.g. jobs) per month. Many task computing denotes high-performance computations comprising multiple distinct activities, coupled via file system operations. Tasks may be small or large, uniprocessor or multiprocessor, compute-intensive or data-intensive. The set of tasks may be static or dynamic, homogeneous or heterogeneous, loosely coupled or tightly coupled. The aggregate number of tasks, quantity of computing, and volumes of data may be extremely large. Many task computing includes loosely coupled applications that are generally communication-intensive but not naturally expressed using standard message passing interface commonly found in high performance computing, drawing attention to the many computations that are heterogeneous but not "happily" parallel.

Keywords: many-task computing, MTC, high-throughput computing, HTC, high performance computing, HPC

1. Defining Many Task Computing

We want to enable the use of large-scale distributed systems for task-parallel applications, which are linked into useful workflows through the looser task-coupling model of passing data via files between dependent tasks. This potentially larger class of task-parallel applications is precluded from leveraging the increasing power of modern parallel systems such as supercomputers (e.g. IBM Blue Gene/L [1] and Blue Gene/P [2]) because the lack of efficient support in those systems for the "scripting" programming model [3]. With advances in e-Science and the growing complexity of scientific analyses, more scientists and researchers rely on various forms of scripting to automate end-to-end application processes involving task coordination, provenance tracking, and bookkeeping. Their approaches are typically based on a model of loosely coupled computation, in which data is exchanged among tasks via files, databases or XML documents, or a combination of these. Vast increases in data volume combined with the growing complexity of data analysis procedures and algorithms have rendered traditional manual processing and exploration unfavorable as compared with modern high performance computing processes automated by scientific workflow systems. [4]

The problem space can be partitioned into four main categories (Figure 1 and Figure 2). 1) At the low end of the spectrum (low number of tasks and small input size), we have tightly coupled Message Passing Interface (MPI) applications (white). 2) As the data size increases, we move into the analytics category, such as data mining and analysis (blue); MapReduce [5] is an example for this category. 3) Keeping data size modest, but increasing the number of tasks moves us into the loosely coupled applications involving many tasks (yellow); Swift/Falkon [6, 7] and Pegasus/DAGMan [8] are examples of this category. 4) Finally, the combination of both many tasks and large datasets moves us into the data-intensive many-task computing category (green); examples of this category

978-1-4244-2872-4/08/$25.00

are Swift/Falkon and data diffusion [9], Dryad [10], and Sawzall [11].

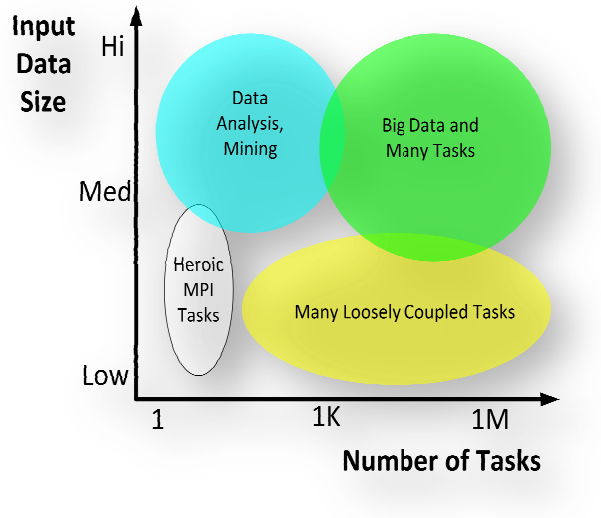

Figure 1: Problem types with respect to data size and number of tasks

High performance computing can be considered to be part of the first category (denoted by the white area). High throughput computing [12] can be considered to be a subset of the third category (denoted by the yellow area). Many-Task Computing can be considered as part of categories three and four (denoted by the yellow and green areas). This paper focuses on defining many-task computing, and the challenges that arise as datasets and computing systems are growing exponentially.

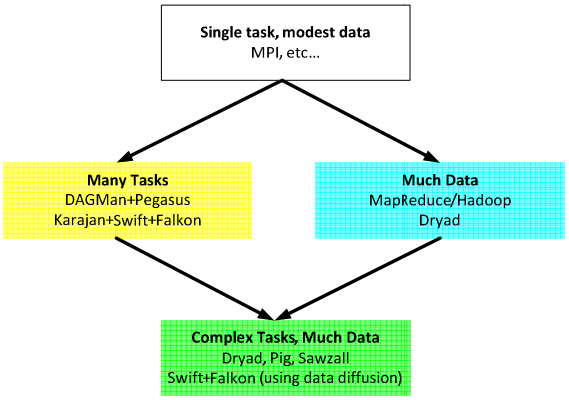

Figure 2: An incomplete and simplistic view of programming models and tools

Clusters and Grids have been the preferred platform for loosely coupled applications that have been traditionally part of the high throughput computing class of applications, which are managed and executed through workflow systems or parallel programming systems. Various properties of a new emerging applications, such as large number of tasks (i.e. millions or more), relatively short per task execution times (i.e. seconds to minutes long), and data intensive tasks (i.e. tens of MB of I/O per CPU second of compute) have lead to the definition of a new class of applications called Many-Task Computing. MTC emphasizes on using much large numbers of computing resources over short periods of time to accomplish many computational tasks, where the primary metrics are in seconds (e.g., FLOPS, tasks/sec, MB/sec I/O rates), while HTC requires large amounts of computing for long periods of time with the primary metrics being operations per month [12]. MTC applications are composed of many tasks (both independent and dependent tasks) that can be individually scheduled on many different computing resources across multiple administrative boundaries to achieve some larger application goal.

MTC denotes high-performance computations comprising multiple distinct activities, coupled via file system operations or message passing. Tasks may be small or large, uniprocessor or multiprocessor, compute-intensive or data-intensive. The set of tasks may be static or dynamic, homogeneous or heterogeneous, loosely coupled or tightly coupled. The aggregate number of tasks, quantity of computing, and volumes of data may be extremely large. Is MTC really different enough to justify coining a new term? There are certainly other choices we could have used instead, such as multiple program multiple data (MPMD), high throughput computing, workflows, capacity computing, or embarrassingly parallel.

MPMD is a variant of Flynn's original taxonomy [13], used to denote computations in which several programs each operate on different data at the same time. MPMD can be contrasted with Single Program Multiple Data (SPMD), in which multiple instances of the same program each execute on different processors, operating on different data. MPMD lacks the emphasis that a set of tasks can vary dynamically. High throughput computing [12], a term coined by Miron Livny within the Condor project [14], to contrast workloads for which the key metric is not floating-point operations per second (as in high performance computing) but "per month or year." MTC applications are often just as concerned with performance as is the most demanding HPC application; they just don't happen to be SPMD programs. The term "workflow" was first used to denote sequences of tasks in business processes, but the term is sometimes used to denote any computation in which control and data passes from one "task" to another. We find it often used to describe many-task computations (or MPMD, HTC, MTC, etc.), making its use too general. "Embarrassingly parallel computing" is

used to denote parallel computations in which each individual (often identical) task can execute without any significant communication with other tasks or with a file system. Some MTC applications will be simple and embarrassingly parallel, but others will be extremely complex and communication-intensive, interacting with other tasks and shared file-systems.

Is "many task computing" a useful distinction? Perhaps we could simply have said "applications that are communication-intensive but are not naturally expressed in MPI", but there are also loosely coupled, and independent many tasks. Through the new term MTC, we are drawing attention to the many computations that are heterogeneous but not "happily" parallel.

2. MTC for Clusters, Grids, and Supercomputers

We claim that MTC applies to not only traditional HTC environments such as clusters and Grids, assuming appropriate support in the middleware, but also supercomputers. Emerging petascale computing systems, such as IBM's Blue Gene/P [2], incorporate high-speed, low-latency interconnects and other features designed to support tightly coupled parallel computations. Most of the applications run on these computers have a SMPD structure, and are commonly implemented by using MPI to achieve the needed inter-process communication. We believe MTC to be a viable paradigm for supercomputers. As the computing and storage scale increases, the set of problems that must be overcome to make MTC practical (ensuring good efficiency and utilization at large-scale) exacerbate. The challenges include local resource manager scalability and granularity, efficient utilization of the raw hardware, shared file system contention and scalability, reliability at scale, application scalability, and understanding the limitations of the HPC systems in order to identify promising and scientifically valuable MTC applications.

One could ask, why use petascale systems for problems that might work well on terascale systems? We point out that petascale systems are more than just many processors with large peak petaflop ratings. They normally come well balanced, with proprietary, high-speed, and low-latency network interconnects to give tightly-coupled applications good opportunities to scale well at full system scales. Even IBM has proposed in their internal project Kittyhawk [15] that Blue Gene/P can be used to run non-traditional workloads (e.g. HTC). We identify four factors that motivate the support of MTC applications on petascale HPC systems:

1) *The I/O subsystems of petascale systems offer unique capabilities needed by MTC applications.* For example, collective I/O operations [16] could be implemented to use the specialized high-bandwidth and low-latency interconnects. MTC applications could be composed of individual tasks that are themselves parallel programs, many tasks operating on the same input data, and tasks that need considerable communication among them. Furthermore, the aggregate shared file system performance of a supercomputer can be potentially larger than that found in a distributed infrastructure (i.e., Grid), with data rates in the 10GB+/s range, rather than the more typical 0.1GB/s to 1GB/s range at most Grid sites.

2) *The cost to manage and run on petascale systems like the Blue Gene/P is less than that of conventional clusters or Grids*[15]. For example, a single 13.9 TF Blue Gene/P rack draws 40 kilowatts, for 0.35 GF/watt. Two other systems that get good compute power per watt consumed are the SiCortex with 0.32 GF/watt and the Blue Gene/L with 0.23 GF/watt. In contrast, the average power consumption of the Top500 systems is 0.12 GF/watt [17]. Furthermore, we also argue that it is more cost effective to manage one large system in one physical location, rather than many smaller systems in geographically distributed locations.

3) *Large-scale systems inevitably have utilization issues.* Hence it is desirable to have a community of users who can leverage traditional back-filling strategies to run loosely coupled applications on idle portions of petascale systems.

4) *Perhaps most importantly, some applications are so demanding that only petascale systems have enough compute power to get results in a reasonable timeframe, or to exploit new opportunities in such applications.* With petascale processing capabilities on ordinary applications, it becomes possible to perform vast computations with quick turn-around, thus answering questions in a timeframe that can make a quantitative difference in addressing significant scientific challenges or responding to emergencies.

3. The Data Deluge Challenge and the Growing Storage/Compute Gap

Within the science domain, the data that needs to be processed generally grows faster than computational resources and their speed. The scientific community is facing an imminent flood of data expected from the next generation of experiments, simulations, sensors

and satellites. Scientists are now attempting calculations requiring orders of magnitude more computing and communication than was possible only a few years ago. Moreover, in many currently planned and future experiments, they are also planning to generate several orders of magnitude more data than has been collected in the entire human history [18].

For instance, in the astronomy domain the Sloan Digital Sky Survey [19] has datasets that exceed 10 terabytes in size. They can reach up to 100 terabytes or even petabytes if we consider multiple surveys and the time dimension. In physics, the CMS detector being built to run at CERN's Large Hadron Collider [20] is expected to generate over a petabyte of data per year. In the bioinformatics domain, the rate of growth of DNA databases such as GenBank [21] and European Molecular Biology Laboratory (EMBL) [22] has been following an exponential trend, with a doubling time estimated to be 9-12 months. A large class of applications in Many-Task Computing will be applications that analyze large quantities of data, which in turn would require that data and computations be distributed over many hundreds and thousands of nodes in order to achieve rapid turnaround times.

Many applications in the scientific computing generally use a shared infrastructure such as TeraGrid [23] and Open Science Grid [24], where data movement relies on shared or parallel file systems. The rate of increase in the number of processors per system is outgrowing the rate of performance increase of parallel file systems, which requires rethinking existing data management techniques. For example, a cluster that was placed in service in 2002 with 316 processors has a parallel file system (i.e. GPFS [25]) rated at 1GB/s, yielding 3.2MB/s per processor of bandwidth. The second largest open science supercomputer, the IBM Blue Gene/P from Argonne National Laboratory, has 160K processors, and a parallel file system (i.e. also GPFS) rated at 8GB/s, yielding a mere 0.05MB/s per processor. That is a 65X reduction in bandwidth between a system from 2002 and one from 2008. Unfortunately, this trend is not bound to stop, as advances multi-core and many-core processors will increase the number of processor cores one to two orders of magnitude over the next decade. [4]

We believe that data locality is critical to the successful and efficient use of large distributed systems for data-intensive applications [26, 27] in the face of a growing gap between compute power and storage performance. Large scale data management needs to be a primary objective for any middleware targeting to support MTC workloads, to ensure data movement is minimized by intelligent data-aware scheduling both among distributed computing sites (assuming that each site has a local area network shared storage infrastructure), and among compute nodes (assuming that data can be stored on compute nodes' local disk and/or memory).

4. Middleware Support for MTC

As high throughput computing (HTC) is a subset of MTC, it is worth mentioning the various efforts in enabling HTC on large scale systems. Some of these systems are Condor [14], MapReduce [5], Hadoop [28], and BOINC [29]. MapReduce (including Hadoop) is typically applied to a data model consisting of name/value pairs, processed at the programming language level. Its strengths are in its ability to spread the processing of a large dataset to thousands of processors with minimal expertise in distributed systems; however it often involves the development of custom filtering scripts and does not support "black box" application execution as is commonly found in MTC or HTC applications. BOINC is known to scale well to large number of compute resources, but lacks support for data intensive applications due to the nature of the wide area network deployment BOINC typically has, as well as support for "black box" applications.

On the IBM Blue Gene supercomputer, various works [30, 31] have leveraged the HTC-mode [32] support in Cobalt [33] scheduling system. These works have aimed at integrating their solutions as much as possible in Cobalt; however, it is not clear that the current implementations will be able to support the largest and most demanding MTC applications at full system scales. Furthermore, these works focus on compute resource management, and ignore data management altogether.

Condor and glide-ins [34] are the original tools to enable HTC, but their emphasis on robustness and recoverability limits their efficiency for MTC applications in large-scale systems. We found that relaxing some constraints from the middleware and encouraging the end applications to implement these constraints have enabled significant improvements in middleware performance and efficiency at large scale. [7]

For example, some of our work with the Falkon light-weight task execution framework has shown that task granularity can be efficiently handled at scales of seconds on modest size clusters of thousands of processing cores, while prior resource management techniques would have required task granularity to be in minutes, or even hours, to allow good utilization of the raw resources at the same scale. The task granularity issues are exaggerated as system scales are increased to

levels of hundreds of thousands of processor cores. Falkon is able to achieve these improvements by streamlining the dispatching process, making the scheduler run in constant time in relation to the number of tasks queued and number of processors managed, multi-threading the resource manager to leverage multi-core nodes, moving recoverability from the resource manager to the application, and leaving out many non-essential features (i.e. accountability, support for multi-node tasks such as MPI, etc) from production managers that can be offloaded to other systems or the application. [7]

Swift [6, 35] and Falkon [7] have been used to execute MTC applications on clusters, multi-site Grids (e.g., Open Science Grid [24], TeraGrid [36]), specialized large machines (SiCortex [37]), and supercomputers (e.g., Blue Gene/P [2]). Swift enables scientific workflows through a data-flow-based functional parallel programming model. It is a parallel scripting tool for rapid and reliable specification, execution, and management of large-scale science and engineering workflows. The runtime system in Swift relies on the CoG Karajan [38] workflow engine for efficient scheduling and load balancing, and it integrates with the Falkon light-weight task execution dispatcher for optimized task throughput and efficiency, as well as improved data management capabilities to ensure good scalability with increasing compute resources. Large-scale applications from many domains (e.g., astronomy [7, 39], medicine [7, 40, 41], chemistry [42], molecular dynamics [43], and economics [44, 45]) have been run at scales of up to millions of tasks on up to hundreds of thousands of processors. While this paper strives to define MTC, as well the theory and practice to enable MTC on a wide range of systems from the average cluster to the largest supercomputers, the Falkon middleware represents the practical aspects of enabling MTC workloads on these systems.

5. MTC Applications

We have found many applications that are a better fit for MTC than HTC or HPC. Their characteristics include having a large number of small parallel jobs, a common pattern observed in many scientific applications [6]. They also use files (instead of messages, as in MPI) for intra-processor communication, which tends to make these applications data intensive.

While we can push hundreds or even thousands of such small jobs via GRAM to a traditional local resource manager (e.g. PBS [46], Condor [34], SGE [47]), the achieved utilization of a modest to large resource set will be poor due to high queuing and dispatching overheads, which ultimately results in low job throughput. A common technique to amortize the costs of the local resource management is to "cluster" multiple jobs into a single larger job. Although this lowers the per job overhead, it is best suited when the set of jobs to be executed are homogenous in execution times, or accurate execution time information is available prior to job execution; with heterogeneous job execution times, it is hard to maintain good load balancing of the underlying resource, causing low resource utilization. We claim that "clustering" jobs is not enough, and that the middleware that manages jobs must be streamlined and made as light-weight as possible to allow applications with heterogonous execution times to execute without "clustering" with high efficiency.

In addition to streamlined task dispatching, scalable data management techniques are also required in order to support MTC applications. MTC applications are often data and/or meta-data intensive, as each job requires at least one input file and one output file, and can sometimes involve many files per job. These data management techniques need to make good utilization of the full network bandwidth of large scale systems, which is a function of the number of nodes and networking technology employed, as opposed to the relatively small number of storage servers that are behind a parallel file system or GridFTP server.

We have identified various applications (as detailed below) from many disciplines that demonstrate characteristics of MTC applications. These applications cover a wide range of domains, from astronomy, physics, astrophysics, pharmaceuticals, bioinformatics, biometrics, neuroscience, medical imaging, chemistry, climate modeling, economics, and data analytics. They often involve many tasks, ranging from tens of thousands to billions of tasks, and have a large variance of task execution times ranging from hundreds of milliseconds to hours. Furthermore, each task is involved in multiple reads and writes to and from files, which can range in size from kilobytes to gigabytes. These characteristics made traditional resource management techniques found in HTC inefficient; also, although some of these applications could be coded as HPC applications, due to the wide variance of the arrival rate of tasks from many users, an HPC implementation would also yield poor utilization. Furthermore, the data intensive nature of these applications can quickly saturate parallel file systems at even modest computing scales.

Astronomy: One of the first applications that motivated much of this work was called the

"AstroPortal" [48], which offered a stacking service of astronomy images from the Sloan Digital Sky Survey (SDSS) dataset using grid resources. Astronomical image collections usually cover an area of sky several times (in different wavebands, different times, etc). On the other hand, there are large differences in the sensitivities of different observations: objects detected in one band are often too faint to be seen in another survey. In such cases we still would like to see whether these objects can be detected, even in a statistical fashion. There has been a growing interest to re-project each image to a common set of pixel planes, then stacking images. The stacking improves the signal to noise, and after coadding a large number of images, there will be a detectable signal to measure the average brightness/shape etc of these objects. This application involved the SDSS dataset [19] (currently at 10TB with over 300 million objects, but these datasets could be petabytes in size if we consider multiple surveys in both time and space) [4], many tasks ranging from 10K to millions of tasks, each requiring 100ms to seconds of compute and 100KB to MB of input and output data.

Another related application in astronomy is MONTAGE [49, 50], a national virtual observatory project [51] that stitches tiles of images of the sky from various sky surveys (e.g. SDSS [19], etc) into a photorealistic single image. Execution times per task range in the 100ms to 10s of seconds, and each task involves multiple input images and at least one image output. This application is both compute intensive and data intensive, and has been run as both a HTC (using Pegasus/DAGMan [8], and Condor [14]) and a HPC (using MPI) application, but we found its scalability to be limited when run under HTC or HPC.

Astrophysics: Another application is from astrophysics, which analyzes the Flash turbulence dataset (simulation data) [52] from various perspectives, using volume rendering and vector visualization. The dataset is composed of 32 million files (1000 time steps times 32K files) taking up about 15TB of storage resource, and contains both temporal and spatial locality. In the physics domain, the CMS detector being built to run at CERN's Large Hadron Collider [20] is expected to generate over a petabyte of data per year. Supporting applications that can perform a wide range of analysis of the LHC data will require novel support for data intensive applications.

Economic Modeling: An application from the economic modeling domain that we have investigated as a good MTC candidate is Macro Analysis of Refinery Systems (MARS) [44], which studies economic model sensitivity to various parameters. MARS models the economic and environmental impacts of the consumption of natural gas, the production and use of hydrogen, and coal-to-liquids co-production, and seeks to provide insights into how refineries can become more efficient through the capture of waste energy. Other economic modeling applications perform numerical optimization to determine optimal resource assignment in energy problems. This application is challenging as the parameter space is extremely large, which can produce millions, even billions of individual tasks, each with a relatively short execution time of only seconds long.

Pharmaceutical Domain: In the pharmaceutical domain, there are applications that screen KEGG [53] compounds and drugs against important metabolic protein targets using DOCK6 [43] to simulate the "docking" of small molecules, or ligands, to the "active sites" of large macromolecules of known structure called "receptors". A compound that interacts strongly with a receptor (such as a protein molecule) associated with a disease may inhibit its function and thus act as a beneficial drug. The economic and health benefits of speeding drug development by rapidly screening for promising compounds and eliminating costly dead-ends is significant in terms of both resources and human life. The parameter space is quite large, totaling to more than one billion computations that have a large variance of execution times from seconds to hours, with an average of 10 minutes. The entire parameter space would require over 22,600 CPU years, or over 50 days on a 160K processor Blue Gene/P supercomputer [2]. This application is challenging as there many tasks, each task has a wide range of execution times with little to no prior knowledge about its execution time, and involves significant I/O for each computation as the compounds are typically stored in a database (i.e. 10s to 100s of MB large) and must be read completely per computation.

Chemistry: Another application in the same domain is OOPS [54], which aims to predict protein structure and recognize docking partners. In chemistry, specifically in molecular dynamics, we have an application MolDyn whose goal is to calculate the solvation free energy of ligands and protein-ligand binding energy, with structures obtained from the NIST Chemistry WebBook database [55]. Solvation free energy is an important quantity in Computational Chemistry with a variety of applications, especially in drug discovery and design. These applications have similar characteristics as the DOCK application previously discussed.

Bioinformatics: In bioinformatics, Basic Local Alignment Search Tool (BLAST), is a family of tools for comparing primary biological sequence information (e.g. amino-acid sequences of proteins, nucleotides of

DNA sequences). A BLAST search enables one to compare a query sequence with a library or database of sequences, and identify library sequences that resemble the query sequence above a certain threshold. [56]. Although the BLAST codes have been implemented in both HTC and HPC, they are often both data and compute intensive, requiring multi-GB databases to be read for each comparison (or kept in memory if possible), and each comparison can be done within minutes on a modern processor-core. MTC and its support for data intensive applications are critical in scaling BLAST on large scale resources with thousands to hundreds of thousands of processors.

Neuroscience Domain: In the neuroscience domain, we have the Computational Neuroscience Applications Research Infrastructure (CNARI), which aims to manage neuroscience tools and the heterogeneous compute resources on which they can enable large-scale computational projects in the neuroscience community. The analysis includes the aphasia study, structural equation modeling, and general use of R for various data analysis. [57] The application workloads involve many tasks, relatively short on the order of seconds, and each task containing many small input and output files making the application meta-data intensive at large scale.

Cognitive Neuroscience: The fMRI application is a workflow from the cognitive neuroscience domain with a four-step pipeline, which includes Automated Image Registration (AIR), Preprocessing and stats from NIH and FMRIB (AFNI and FSL), and Statistical Parametric Mapping (SPM2) [58]. An fMRI Run is a series of brain scans called volumes, with a Volume containing a 3D image of a volumetric slice of a brain image, which is represented by an Image and a Header. Each volume can contain hundreds to thousands of images, and with multiple patients, the number of individual analysis tasks can quickly grow. Task execution times were only seconds long, and the input and output files ranged from kilobytes to megabytes in size. This application could run as an HTC one at small scales, but needs MTC support to scale up.

Data Analytics: Data analytics and data mining is a large field that covers many different applications. Here, we outline several applications that fit MTC well. One example is the analysis of log data from millions computations. Another set of applications are ones commonly found in the MapReduce [59] paradigm, namely "sort" and "word count". Both of these applications are essential to World Wide Web search engines, and are challenging at medium to large scale due to their data intensive nature. All three applications involve many tasks, many input files (or many disjoint sub-sections of few files), and are data intensive.

Data Mining: Another set of applications that perform data analysis can be classified in the "All-Pairs" class of applications [60]. These applications aim to understand the behavior of a function on two sets, or to learn the covariance of these sets on a standard inner product. Two common applications in All-Pairs are data mining and biometrics. Data mining is the study of extracting meaning from large data sets; one phase of knowledge discovery is reacting to bias or other noise within a set. Different classifiers work better or worse for varying data, and hence it is important to explore many different classifiers in order to be able to determine which classifier is best for that type of noise on a particular distribution of the validation set.

Biometrics: Biometrics aims to identifying humans from measurements of the body (e.g. photos of the face, recordings of the voice, and measurements of body structure). A recognition algorithm may be thought of as a function that accepts two images (e.g. face) as input and outputs a number between zero and one indicating the similarity between the two input images. The application would then compare all images of a database and create a scoring matrix which can later be easily searched to retrieve the most similar images. These All-Pairs applications are extremely challenging as the number of tasks can rapidly grow in the millions and billions, with each task being hundreds of milliseconds to tens of seconds, with multi-megabyte input data per task.

MPI Ensembles: Finally, another class of applications is managing an ensemble of MPI applications. One example is from the climate modeling domain, which has been studying climate trends and predicting global warming [61], is already implemented as an HPC MPI application. However, the current climate models could be run as ensemble runs (many concurrent MPI applications) to quantify climate model uncertainty. This is challenging in large scale systems such as supercomputers (a typical resource such models would execute on), as the local resource managers (e.g. Cobalt) favor large jobs and have policy against running many jobs at the same time (i.e. where many is more than single digit number of jobs per user).

All these applications pose significant challenges to traditional resource management found in HPC and HTC, from both job management and storage management perspective, and are in critical need of MTC support as the scale of these resources grows.

6. Conclusions

We have defined a new paradigm – MTC – which aims to bridge the gap between two computing paradigms, HTC and HPC. MTC applications are typically loosely coupled that are communication-intensive but not naturally expressed using standard message passing interface commonly found in high performance computing, drawing attention to the many computations that are heterogeneous but not "happily" parallel. We believe that today's existing HPC systems are a viable platform to host MTC applications. We also believe MTC is a broader definition than HTC, allowing for finer grained tasks, independent tasks as well as ones with dependencies, and allowing tightly coupled applications and loosely coupled applications to co-exist on the same system.

Furthermore, having native support for data intensive applications is central to MTC, as there is a growing gap between storage performance of parallel file systems and the amount of processing power. As the size of scientific data sets and the resources required for analysis increase, data locality becomes crucial to the efficient use of large scale distributed systems for scientific and data-intensive applications [62]. We believe it is feasible to allocate large-scale computational resources and caching storage resources that are relatively remote from the original data location, co-scheduled together to optimize the performance of entire data analysis workloads which are composed of many loosely coupled tasks.

We identified challenges in running these novel workloads on large-scale systems, which can hamper both efficiency and utilization. These challenges include local resource manager scalability and granularity, efficient utilization of the raw hardware, shared file system contention and scalability, reliability at scale, application scalability, and understanding the limitations of HPC systems in order to identify promising and scientifically valuable MTC applications.

Clusters with 62K processor cores (e.g., TACC Sun Constellation System, Ranger), Grids (e.g., TeraGrid) with over a dozen sites and 161K processors, and supercomputers with 160K processors (e.g., IBM Blue Gene/P) are now available to the scientific community. These large HPC systems are considered efficient at executing tightly coupled parallel jobs within a particular machine using MPI to achieve inter-process communication. We proposed using HPC systems for loosely-coupled applications, which involve the execution of independent, sequential jobs that can be individually scheduled, and using files for inter-process communication.

We have already shown good support for MTC on a variety of resources from clusters, grids, and supercomputers through our work on Swift [6, 35, 42] and Falkon [7, 63]. Furthermore, we have taken the first steps to address data-intensive MTC by offloading much of the I/O away from parallel file systems and into the network, making full utilization of caches (both on disk and in memory) and the full network bandwidth of commodity networks (e.g. gigabit Ethernet) as well as proprietary and more exotic networks (Torus, Tree, and Infiniband). [9, 16]

We believe that there is more to HPC than tightly coupled MPI, and more to HTC than embarrassingly parallel long running jobs. Like HPC applications, and science itself, applications are becoming increasingly complex opening new doors for many opportunities to apply HPC in new ways if we broaden our perspective. We hope the definition of Many-Task Computing leads to a stronger appreciation of the fact that applications that are not tightly coupled via MPI are not necessarily embarrassingly parallel: some have just so many simple tasks that managing them is hard, some operate on or produce large amounts of data that need sophisticated data management in order to scale. There also exist applications that involve MPI ensembles, essentially many jobs where each job is composed of tightly coupled MPI tasks, and there are loosely coupled applications that have dependencies among tasks, but typically use files for inter-process communication. Efficient support for these sorts of applications on existing large scale systems, including future ones will involve substantial technical challenges and will have big impact on science.

To extend the discussion of this paper with the broader community on many-task computing, we held a workshop at IEEE/ACM Supercomputing 2008 titled "IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08)" [64] as well as a bird-of-feather (BOF) session titled "Megajobs08: How to Run a Million Jobs" [65]. The half-day workshop was a success; it attracted over 100 participants who attended the presentations of the six accepted papers [16, 66, 67, 68, 69, 70] and a keynote talk by the renowned Dr. Alan Gara (IBM Blue Gene Chief Architect). The BOF was also a success, having seven short presentations and attracting over 60 participants. We plan to have follow-up workshops and special issue publications on many-task computing in the near future.

7. Acknowledgements

This work was supported in part by the NASA Ames Research Center GSRP grant number NNA06CB89H, the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357, and the National Science Foundation under grant OCI-0721939. We also thank the Argonne Leadership Computing Facility for allowing us to test MTC in practice on the IBM Blue Gene/P. We also thank our many colleagues for their contributions to the implementations and applications of making many task computing a reality; in no particular order, we would like to thank Michael Wilde, Zhao Zhang, Allan Espinosa, Ben Clifford, Kamil Iskra, Pete Beckman, Mike Kubal, Don Hanson, Matthew Cohoon, Fangfang Xia, Rick Stevens, Douglas Thain, Matei Ripeanu, and Samer Al-Kiswany.

8. References

[1] A. Gara, et al. “Overview of the Blue Gene/L system architecture”, IBM Journal of Research and Development 49(2/3), 2005

[2] IBM BlueGene/P (BG/P), http://www.research.ibm.com/bluegene/, 2008

[3] J. Ousterhout, “Scripting: Higher Level Programming for the 21st Century”, IEEE Computer, March 1998

[4] Y. Zhao, I. Raicu, I. Foster. “Scientific Workflow Systems for 21st Century e-Science, New Bottle or New Wine?”, IEEE Workshop on Scientific Workflows 2008

[5] J. Dean, S. Ghemawat. “MapReduce: Simplified data processing on large clusters.” In OSDI, 2004

[6] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, I. Raicu, T. Stef-Praun, M. Wilde. “Swift: Fast, Reliable, Loosely Coupled Parallel Computation”, IEEE Workshop on Scientific Workflows 2007

[7] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde. “Falkon: A Fast and Lightweight Task Execution Framework”, IEEE/ACM SC, 2007

[8] E. Deelman et al. “Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems”, Scientific Programming Journal 13(3), 219-237, 2005

[9] I. Raicu, Y. Zhao, I. Foster, A. Szalay. “Accelerating Large-Scale Data Exploration through Data Diffusion”, ACM International Workshop on Data-Aware Distributed Computing 2008

[10] M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly. “Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks”, European Conference on Computer Systems (EuroSys), 2007

[11] R. Pike, S. Dorward, R. Griesemer, S. Quinlan. “Interpreting the Data: Parallel Analysis with Sawzall”, Scientific Programming Journal, Special Issue on Grids and Worldwide Computing Programming Models and Infrastructure 13(4), pp. 227-298, 2005

[12] M. Livny, J. Basney, R. Raman, T. Tannenbaum. “Mechanisms for High Throughput Computing”, SPEEDUP Journal 1(1), 1997

[13] M. Flynn. “Some Computer Organizations and Their Effectiveness”, IEEE Trans. Comput. C-21, pp. 948, 1972

[14] D. Thain, T. Tannenbaum, M. Livny, “Distributed Computing in Practice: The Condor Experience”, Concurrency and Computation: Practice and Experience 17(2-4), pp. 323-356, 2005

[15] J. Appavoo, V. Uhlig, A. Waterland. “Project Kittyhawk: Building a Global-Scale Computer”, ACM Sigops Operating System Review, 2008

[16] Z. Zhang, A. Espinosa, K. Iskra, I. Raicu, I. Foster, M. Wilde. “Design and Evaluation of a Collective I/O Model for Loosely-coupled Petascale Programming”, IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08) 2008

[17] Top500, June 2008, http://www.top500.org/lists/2008/06, 2008

[18] T. Hey, A. Trefethen. “The data deluge: an e-sicence perspective”, Gid Computing: Making the Global Infrastructure a Reality, Wiley, 2003

[19] SDSS: Sloan Digital Sky Survey, http://www.sdss.org/, 2008

[20] CERN’s Large Hadron Collider, http://lhc.web.cern.ch/lhc, 2008

[21] GenBank, http://www.psc.edu/general/software/packages/genbank2008

[22] European Molecular Biology Laboratory, http://www.embl.org, 2008

[23] C. Catlett, et al. "TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications", HPC 2006

[24] Open Science Grid (OSG), http://www.opensciencegrid.org/, 2008

[25] F. Schmuck and R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters", FAST 2002

[26] A. Szalay, A. Bunn, J. Gray, I. Foster, I. Raicu. "The Importance of Data Locality in Distributed Computing Applications", NSF Workflow Workshop 2006

[27] J. Gray. "Distributed Computing Economics", Technical Report MSR-TR-2003-24, Microsoft Research, Microsoft Corporation, 2003

[28] A. Bialecki, M. Cafarella, D. Cutting, O. O'Malley. "Hadoop: A Framework for Running Applications on Large Clusters Built of Commodity Hardware", http://lucene.apache.org/hadoop/, 2005

[29] D.P. Anderson, "BOINC: A System for Public-Resource Computing and Storage", IEEE/ACM Workshop on Grid Computing, 2004

[30] J. Cope, M. Oberg, H.M. Tufo, T. Voran, M. Woitaszek. "High Throughput Grid Computing with an IBM Blue Gene/L", Cluster 2007

[31] A. Peters, A. King, T. Budnik, P. McCarthy, P. Michaud, M. Mundy, J. Sexton, G. Stewart. "Asynchronous Task Dispatch for High Throughput Computing for the eServer IBM Blue Gene® Supercomputer", Parallel and Distributed Processing (IPDPS), 2008

[32] IBM Corporation. "High-Throughput Computing (HTC) Paradigm", IBM System Blue Gene Solution: Blue Gene/P Application Development, IBM RedBooks, 2008

[33] N. Desai. "Cobalt: An Open Source Platform for HPC System Software Research", Edinburgh BG/L System Software Workshop, 2005

[34] J. Frey, T. Tannenbaum, I. Foster, M. Frey, S. Tuecke. "Condor-G: A Computation Management Agent for Multi-Institutional Grids", Cluster Computing, 2002

[35] Swift Workflow System, www.ci.uchicago.edu/swift, 2008

[36] C. Catlett et al., "TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications", HPC and Grids in Action, ed. Lucio Grandinetti, IOS Press Advances in Parallel Computing series, Amsterdam, 2007

[37] SiCortex, http://www.sicortex.com/, 2008

[38] G.v. Laszewski, M. Hategan, D. Kodeboyina. "Java CoG Kit Workflow", in I.J. Taylor, E. Deelman, D.B. Gannon, and M. Shields, eds., Workflows for eScience, pp. 340-356, 2007

[39] J.C. Jacob, et al. "The Montage Architecture for Grid-Enabled Science Processing of Large, Distributed Datasets", Earth Science Technology Conference 2004

[40] The Functional Magnetic Resonance Imaging Data Center, http://www.fmridc.org/, 2007

[41] T. Stef-Praun, B. Clifford, I. Foster, U. Hasson, M. Hategan, S. Small, M. Wilde and Y. Zhao. "Accelerating Medical Research using the Swift Workflow System", Health Grid , 2007

[42] Y. Zhao, I. Raicu, I. Foster, M. Hategan, V. Nefedova, M. Wilde. "Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments", Grid Computing Research Progress, Nova Pub. 2008

[43] D.T. Moustakas et al. "Development and Validation of a Modular, Extensible Docking Program: DOCK 5", J. Comput. Aided Mol. Des. 20, pp. 601-619, 2006

[44] D. Hanson. "Enhancing Technology Representations within the Stanford Energy Modeling Forum (EMF) Climate Economic Models", Energy and Economic Policy Models: A Reexamination of Fundamentals, 2006

[45] T. Stef-Praun, G. Madeira, I. Foster, R. Townsend. "Accelerating solution of a moral hazard problem with Swift", e-Social Science, 2007

[46] B. Bode, D.M. Halstead, R. Kendall, Z. Lei, W. Hall, D. Jackson. "The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters", Usenix, 4th Annual Linux Showcase & Conference, 2000

[47] W. Gentzsch, "Sun Grid Engine: Towards Creating a Compute Power Grid", 1st International Symposium on Cluster Computing and the Grid, 2001

[48] I. Raicu, I. Foster, A. Szalay, G. Turcu. "AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis", TeraGrid Conference 2006

[49] G.B. Berriman, et al., "Montage: a Grid Enabled Engine for Delivering Custom Science-Grade Image Mosaics on Demand", SPIE Conference on Astronomical Telescopes and Instrumentation. 2004

[50] J.C. Jacob, et al. "The Montage Architecture for Grid-Enabled Science Processing of Large, Distributed Datasets", Earth Science Technology Conference 2004

[51] US National Virtual Observatory (NVO), http://www.us-vo.org/index.cfm, 2008

[52] ASC / Alliances Center for Astrophysical Thermonuclear Flashes, http://www.flash.uchicago.edu/website/home/, 2008

[53] KEGG's Ligand Database: http://www.genome.ad.jp/kegg/ligand.html, 2008

[54] PL protein library, http://protlib.uchicago.edu/, 2008

[55] NIST Chemistry WebBook database, http://webbook.nist.gov/chemistry/, 2008

[56] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman. "Basic Local Alignment Search Tool", J Mol Biol 215 (3): 403–410, 1990

[57] Computational Neuroscience Applications Research Infrastructure, http://www.ci.uchicago.edu/wiki/bin/view/CNARI/WebHome, 2008

[58] The Functional Magnetic Resonance Imaging Data Center, http://www.fmridc.org/, 2007

[59] J. Dean and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters", Symposium on Operating System Design and Implementation (OSDI'04), 2004

[60] C. Moretti, J. Bulosan, D. Thain, and P. Flynn. "All-Pairs: An Abstraction for Data-Intensive Cloud Computing", IPDPS 2008

[61] D. Bernholdt, S. Bharathi, D. Brown, K. Chanchio, M. Chen, A. Chervenak, L. Cinquini, B. Drach, I. Foster, P. Fox, J. Garcia, C. Kesselman, R. Markel, D. Middleton, V. Nefedova, L. Pouchard, A. Shoshani, A. Sim, G. Strand, and D. Williams, "The Earth System Grid: Supporting the Next Generation of Climate Modeling Research", Proceedings of the IEEE, 93 (3), p 485-495, 2005

[62] A. Szalay, A. Bunn, J. Gray, I. Foster, I. Raicu. "The Importance of Data Locality in Distributed Computing Applications", NSF Workflow Workshop 2006

[63] I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, B. Clifford. "Towards Loosely-Coupled Programming on Petascale Systems", IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SuperComputing/SC08), 2008

[64] I. Raicu, Y. Zhao, I. Foster. "IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08) 2008", co-located with IEEE/ACM Supercomputing 2008, http://dsl.cs.uchicago.edu/MTAGS08/, 2008

[65] M. Pierce, I. Raicu, R. Pordes, J. McGee, D. Repasky. "How to Run One Million Jobs (Megajobs08)", Bird-of-Feather Session at IEEE/ACM Supercomputing 2008, http://gridfarm007.ucs.indiana.edu/megajobBOF/index.php/Main_Page, 2008

[66] Y. Gu, R. Grossman. "Exploring Data Parallelism and Locality in Wide Area Networks", IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08) 2008

[67] E.V. Hensbergen, Ron Minnich. "System Support for Many Task Computing", IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08) 2008

[68] L. Hui, Y. Huashan, L. Xiaoming. "A Lightweight Execution Framework for Massive Independent Tasks", IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08) 2008

[69] A.T. Thor, G.V. Zaruba, D. Levine, K. De, T.J. Wenaus. "ViGs: A Grid Simulation and Monitoring Tool for Grid Workflows", IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08) 2008

[70] E. Afgan, P. Bangalore. "Embarrassingly Parallel Jobs Are Not Embarrassingly Easy to Schedule on the Grid", IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08) 2008