High Throughput Landmark Based Image Registration Using Cloud Computing

Lin Yang[1,2], Hyunjoo Kim[3], Manish Parashar[3], and David J. Foran[1,2]

[1] Dept. of Radiology, UMDNJ-Robert Wood Johnson Medical School
[2] Center of Biomedical Imaging and Informatics, The Cancer Institute of New Jersey, UMDNJ-Robert Wood Johnson Medical School, Piscataway, NJ, 08854, USA
[3] Dept. of Electrical and Computer Eng., Rutgers Univ., Piscataway, NJ, 08544, USA

Abstract. Cloud computing provides an elastic, as-a-Service resource abstraction with a usage-based payment model, and is emerging as an attracting computing paradigm. This is especially true for computing intensive application workflows with high-throughput requirements, such as those involving large-scale image datasets. In this paper, we investigate the usage of cloud computing to support a medical image registration workflow using the CometCloud autonomic cloud computing framework. CometCloud supports the execution on workflows on heterogeneous, dynamically federated (private and public) clouds, where the federation is driven by application requirements and user objectives. Specifically, in this investigation, we use objectives such as workflow execution deadlines and budget constraints to provision the appropriate type and number of resources, which are the federated with local private resources to provide additional resources to satisfy the objectives. We experimentally tested this cloud-based workflow on a large dataset containing two sets of consecutive digitized breast tissue specimens. This cloud computing based image registration workflow shows promising results in its ability to meet high-throughput, deadline driven requirements.

1 Introduction

A set of image acquired at different time, or from different perspectives, will be in different coordinate systems. It is therefore critical to align those images into the same coordinate system before applying any following image analysis. Image registration can be separated as rigid registration [1,2] and non-rigid registration [3,4]. Given the fixed and moving images, both problems can be described as finding a linear/nonlinear transformation which maps each point in the fixed image to a point in the moving image. There are usually two ways to describe the transformations. The first method is to define a dense nonparametric model and estimate the position of each point after the registration. Many nonrigid registration algorithm based on elastic deformations, such as fluid deformation based algorithm [5,6] and Demon's algorithm [7,8], fall into this category. The second approach is to model the transformation by a function with some parameters. By estimating those parameters, the algorithm can register each point in the

fixed image to the moving image. B-spline based image registration algorithm belongs to this group. The modeling of the nonlinear deformation using B-spline was introduced in [9]. Mutual information was used in [10] as the cost function where the transformation is modeled as B-spline. A block nonlinear Gauss-Seidel algorithm was used in [11] to minimize the cost function in 2D image registration, where the transformation is modeled as B-spline. B-spline transformation was extended for 3D image registration in [12].

Public clouds have emerged as an important resource class enabling the renting of resources on-demand, and supporting a pay-as-you-go pricing policy. Furthermore, private clouds or data centers are exploring the possibility of scaling out to public clouds to respond to un-anticipated resource requirements. As a result, dynamically federated, hybrid cloud infrastructures that integrate enterprise datacenters, grids, private and/or public clouds are becoming increasingly important. Such federated cloud infrastructures also provide opportunities to improve application quality of service by allowing applications tasks to be mapped to appropriate resource classes. For example, a typical application workflow consists of multiple application stages, which in turn can be composed of different application components with heterogeneous computing requirement in terms of the complexity of the tasks, their execution time, as well as their data requirements. Managing and optimizing these workflows on dynamically federated hybrid clouds can be challenging, especially since it requires simultaneously addressing resource provisioning, scheduling and mapping while balancing QoS with costs.

In this paper, we will present a fast and simple image registration algorithm, and our major focus is its cloud computing based implementation. The algorithm starts from an automatic detection of the landmarks followed by a coarse to fine estimation of the nonlinear mapping. Multi-resolution orientation histograms is used to obtain fast and dense local descriptor of the detected landmarks. Because there is a large portion of outliers in the initial landmark correspondence, a robust estimator, RANSAC [13], is applied to reject outliers. The final refined inliers are used to robustly estimate a thin spline transform (TPS) [14] to complete the final nonlinear registration. The algorithm is simple but also robust enough to handle especially large transformation and deformation. The algorithm is separated into several stages and put into a pipeline structure. This pipeline structure is utilized as a workflow in the cloud computing environment. The workflows are described for each image and deployed on the federated cloud of Rutgers cluster and Amazon EC2 public cloud. We set a user constraint, or deadline, to complete a certain number of pairs of images, then, resources are provisioned from the in-house cluster (Rutgers cluster) first, and *cloudburst* out to public clouds (Amazon EC2) if more resources are required to meet the user constraint. The workflow framework is implemented on top of the CometCloud [15] autonomic cloud engine, which supports dynamic cloud federation, autonomic cloud bursting to scale out to public clouds, and autonomic cloud bridging to integrate multiple datacenters, grids and clouds on-demand. The workflow is fully tested on a large dataset which contains consecutively cut breast tissue speci-

mens which are digitized using standard light microscope with Hematoxylin & Eosin staining.

In Section 2 we introduce the image registration algorithm. In Section 3, we describe the cloud implementation using the CometCloud including the cloud computing experimental results. Finally Section 4 concludes the paper.

2 Image Registration

The registration starts from automatic landmark detection. This step involves the procedure of accurate detection of the prominent or salient points in the image. Harris corner detector is applied to find the point with the large gradient on all the directions (x and y direction for 2D image). After we detect the landmarks, we can extract features from the neighborhood of each landmark. For 2D images, the local orientation histograms are used as the features for landmark matching. The image is first convolved with the orientation filters. The filtering response in the neighborhood around the landmarks is computed to compose the local orientation histogram. The feature vector encodes the directions of the edges at each landmark point and is proven to be quite effective especially when the training samples are small [16].

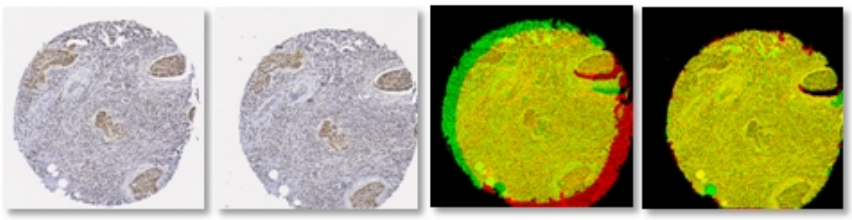
Because the original matching landmark sets between image pairs contain mis-matched landmarks. RANdom SAmple Consensus (RANSAC) [13] is used to reject outliers and robustly estimate the nonlinear transformation. The RANSAC robust estimator randomly chooses a minimal subset of the landmarks to fit the model. Measured by a cost function, the points within a small distance will be considered as a consensus set. The size of the consensus set is called the model support M. The algorithm is repeated multiple times and the model with the largest support is called the robust fit.

The thin plate spline transform (TPS) is finally applied to estimate the nonlinear transformation between the fixed and moving image based on these robust landmark correspondence. By minimizing the bending energy of the mapping transformation T, TPS can provide a smooth matching function for each point in both images. The resulting nonlinear transformation T is utilized to map the moving image to the fixed image.

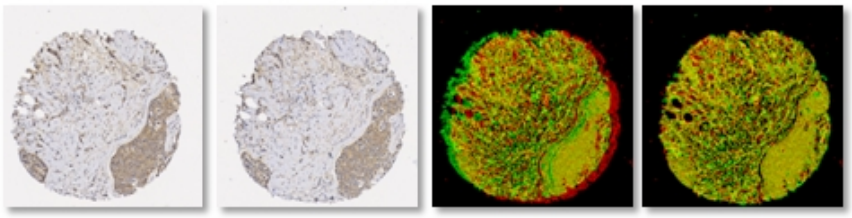
The algorithm is tested on two pairs of consecutively cut slides of breast tissue specimens, which were digitized using standard light microscope. The staining method is standard Hematoxylin & Eosin. We have total 460 image pairs and some registration results are shown in Figure 1.

3 High Throughput Image Registration Using Cloud Computing

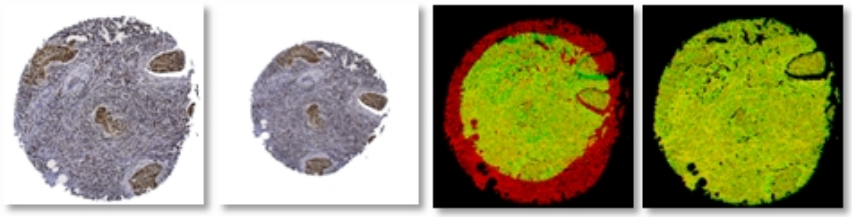
The whole registration algorithm contains both the linear registration and nonlinear registration procedures. Each procedure contains the same steps except the final transformation estimation. The linear affine registration result is served as the initial guess for the nonlinear image registration procedure to avoid the



(a)



(b)



(c)

Fig. 1. The registration results on pathology image specimens: From left to right: the fixed image, the moving image, the difference image before registration and the difference image after registration. Because fixed image and moving image are presented as red and green, respectively, more yellow region represents higher registration results. (a) and (b) are two consecutive breast cancer specimens. (c) Synthetic transformation which contains 45 degree of rotation and 2 times of scale change.

local minimum of the cost function during optimization. In Figure 2 we show the executive time profile for each step in the proposed image registration algorithm. Both linear and nonlinear registration procedures follow similar steps as shown in Figure 2.

In order to utilize the cloud computing framework, we separate the whole registration algorithm into four different stages. The first two stages represent the linear registration part, where the first stage contains the filtering, landmark detection and point matching steps, and the second stage contains the robust estimation and TPS transformation estimation steps. The last two stages contain the nonlinear registration procedure with the same configuration of the steps as linear registration procedure. This proposed pipeline structure by separating the original algorithm into stages can balance the computation-intensive steps

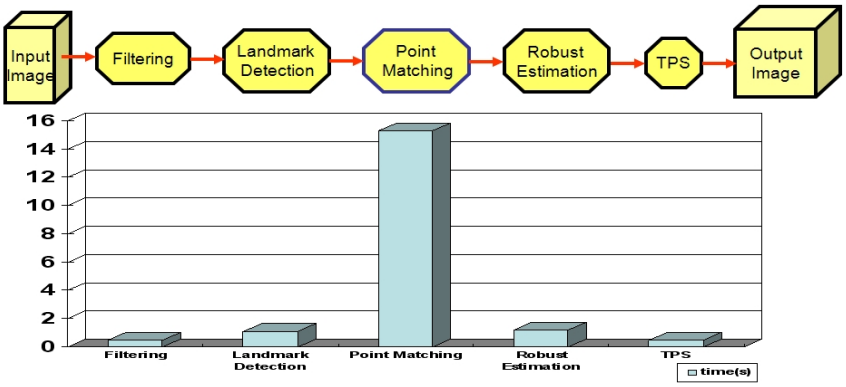


Fig. 2. The time profile for the entire nonlinear registration algorithm

(e. g. point matching), memory intensive step (e. g. landmark detection), or communication-intensive step (e. g. image input and output, filtering, etc.). By carefully staging the image registration algorithm, we also prevent the unnecessary large image data transfer between computational nodes.

3.1 Workflow Framework Using CometCloud

A workflow consists of a sequence of stages and the output of one stage becomes the input of the next stage. Each stage can have a quite heterogeneous resource requirement because stages can run different applications, such as the combination of computation-intensive, memory-intensive, or communication-intensive applications. Even though stages run the same application, each stage can have different number of tasks, variable length of computation, etc. resulting from the previous stage so as to require different resource requirements. Also, as different types of resource classes, such as cluster, datacenter, grid and cloud can be consolidated and build a federated infrastructure, there can be more opportunities to select appropriate resources for heterogeneous applications on-demand on the fly.

The workflow framework is built using CometCloud [17,15], an autonomic computing engine that enables the dynamic and on-demand federation of clouds and grids as well as the deployment and execution of applications on these federated environments. It supports highly heterogeneous and dynamic cloud/grid infrastructures, enabling the integration of public/private clouds and autonomic cloud bursts, i.e., dynamic scale-out to clouds to address dynamic workloads, spikes in demands, and other extreme requirements.

The workflow framework builds a federated cloud and runs user applications on the federated cloud. A cloud can join or leave the federated cloud dynamically to scale up/down. The resource status of nodes in each cloud, such as available CPU, memory, network bandwidth, etc. are monitored and referred to make

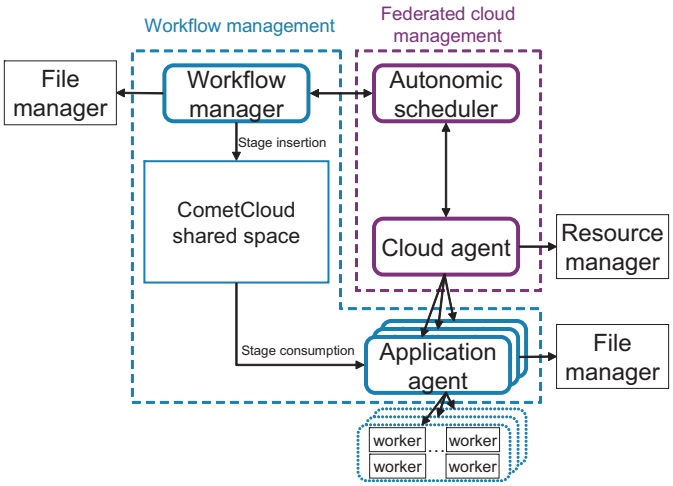


Fig. 3. The architecture of the workflow framework

decisions for scheduling jobs. An application or a sequence of applications described as a workflow is submitted to the workflow framework and consumed on the federated cloud. Typically, a workflow consists of a sequence of applications and the output of one stage becomes the input of the next stage. Hence, each stage should be completed in an order. Furthermore, the workflow framework can support other mostly used master/worker and MapReduce applications by describing the workflow with a single stage.

Figure 3 shows the diagram of the workflow framework. The autonomic scheduler and cloud agents manages the federated cloud, and the workflow manager and application agents are responsible to manage workflows. The workflow manager submits workflows from users into the CometCloud shared space and an application agent picks up a stage from the space so as to consume it in its local cloud. The detailed roles of each component are as follows.

- *Autonomic scheduler*: It is responsible for (1) managing the federated cloud where a cloud dynamically joins or leaves and having a global resource view for available nodes and working nodes, (2) monitoring resource status of cloud nodes through the resource manager, and (3) scheduling stages by selecting the mix of clouds and deciding the number of nodes per cloud based on user objectives, changing resource status and workloads.
- *Resource manager*: It gathers realtime resource status of cloud nodes such as CPU, memory and network bandwidth usage periodically from cloud agents. It provides the resource status of a specific node, a group of nodes, nodes of a cloud, or all nodes in the federated cloud to the autonomic scheduler as requested.
- *Cloud agent*: It is responsible for joining and leaving the federated cloud with a group of cloud nodes. It also manages and monitors local cloud nodes

and resource status. It observes dynamically changing node availability and resource usage during the stage runs.

- *Workflow manager*: When it receives a new workflow, it asks a schedule for the workflow to the autonomic scheduler, submits each stage of the workflow in a sequence, monitors stage progress and receives the results of those stages. Once the workflow is completed, it stores the results until the user retrieves them.
- *Application agent*: It picks up a stage from the CometCloud shared space and consumes it in its local cloud. It asks provisioning of workers to the autonomic scheduler to execute the stage. After it completes the stage, it sends the result to the workflow manager and asks releasing of workers to the autonomic scheduler.
- *File Manager*: Before a stage starts, the relevant input data should be transferred to the cloud node where the stage runs. Also the output data should be gathered in the workflow manager after a stage is completed so that users can retrieve results in a short time. The file managers in the workflow manager and application agents are responsible to transfer input or output data.

3.2 Experiments

We used Rutgers cluster with 20 nodes as a local resource class where each node has 8 cores, 6 GB memory, 146 GB storage and 1 GB Ethernet connection, and Amazon EC2 as a public cloud for cloudburst. We used time constraints, or deadline, as a user objective and cloud resources can be allocated to achieve the user objective if the time constraint is too strict and the local resources are not enough to meet the constraint. To decide the instance type of Amazon EC2 for cloudburst, we first benchmarked the runtime of each stage for an image on several instance types. Table 1 shows that Rutgers cluster node shows better performance than EC2 nodes and c1.medium instance type has the shortest runtime among other EC2 instance types. We did not enable multi-threaded run for multi-core systems, hence, the EC2 instance type which has more cores does not decrease the runtime. Therefore, we used only c1.medium for cloudburst. From this benchmarks, stage 2 and stage4 take less than 1 second, which is too short compared to other stages, hence, we consists of two stages as a workflow combining stage1 and stage2, and stage3 and stage4 instead of four stages.

Table 1. Benchmarks for an image on Rutgers node and EC2 instances.

(seconds)	Rutgers	c1.medium	c1.xlarge	m1.xlarge
stage1	14	17	18	21
stage2	<1	<1	<1	<1
stage3	14	17	18	21
stage4	<1	<1	<1	<1

The autonomic scheduler decides the number of EC2 nodes to achieve the deadline by a simple calculation because the execution time of each image is approximately the same. Let us define R_n, R_t, R_i as the number of Rutgers nodes, the benchmark time for executing an image by a Rutgers node, and the number of images completed by Rutgers nodes, respectively. Also let us define E_n, E_t, E_i as the number of EC2 nodes, the benchmark time of executing an image by an EC2 node, and the number of images completed by EC2 nodes, respectively. Then R_i for a given deadline, T_D, is $R_i = T_D \times R_n/R_t$. If we represent the total number of images to N, then $E_i = N - R_i$ and $E_n = \lceil \frac{(N_i - R_i) \times E_t}{T_D} \rceil$. Therefore, the number of EC2 nodes to be scheduled is

$$E_n = \lceil \frac{N_i}{T_D} - \frac{R_n}{R_t} E_t \rceil \quad (1)$$

Figure 4 shows the result of deadline-driven cloudburst. First, we ran the whole images without cloudburst on 20 Rutgers nodes. The total number of image pairs is 460 and the time-to-completion is 740 seconds as shown as no deadline in the graphs. After then, we varied the deadline from 700 seconds to 200 seconds decreasing by 100 seconds. Figure 4 (a) shows that deadlines are achieved well even though the overhead of transferring output image after a stage does not considered in scheduling the number of nodes. When a worker completes a stage, it puts the output file into the queue of its file server and picks up another stage to execute immediately instead of waiting for the data transfer completed. Note that the workflow manager and each application agent run their own file servers and the file servers are responsible to transfer input/output files. This can overlap data transfer time with image processing time, hence, it increases the total time-to-completion insignificantly. The number of scheduled EC2 nodes is shown in (c) and the total number of executed images completed by Rutgers nodes and EC2 nodes are presented in (b). The figure (d) shows EC2 cost to achieve each deadline limit and it is calculated based on EC2 pricing policy which bills $0.17 hourly for c1.medium instance type.

4 Conclusion

In this paper, we present a cloud computing based image registration application. We propose a deadline driven approach, which shows promising results when running a simple image registration algorithm using cloud services. It is quite clear that cloud computing can play an important role in other high-throughput, large-scale medical image analysis applications.

5 Acknowledgement

This research is funded, in part, by grants from the NIH through contracts 5R01LM009239-04 and 3R01LM009239-03S2 from the National Library of Medicine and contract 9R01CA156386-05A1 from the National Cancer Institute.

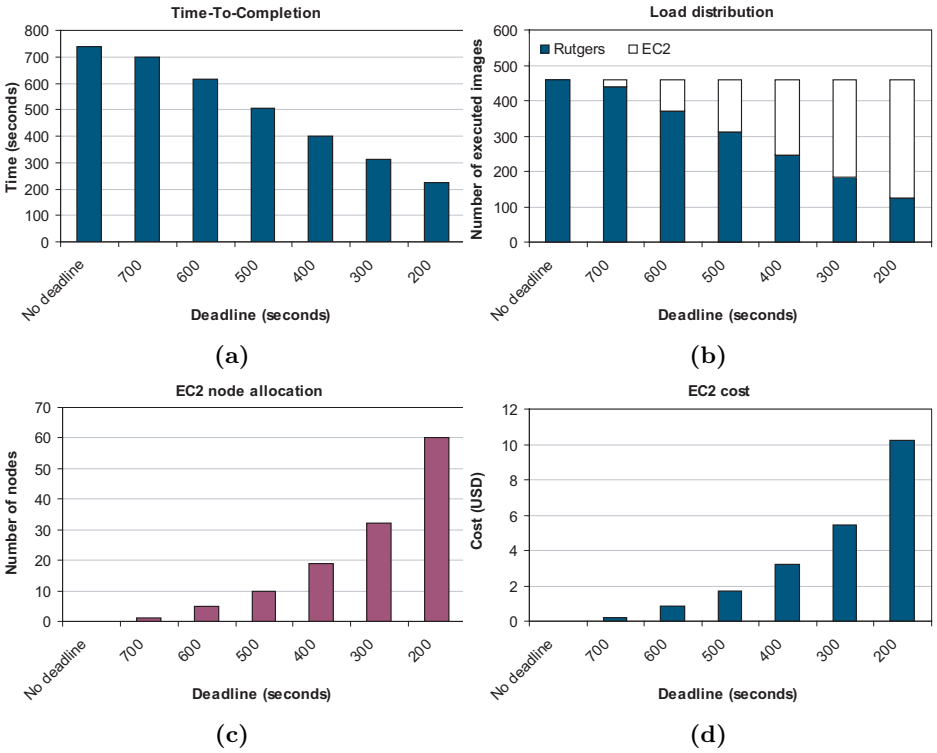


Fig. 4. The result of deadline-driven cloudburst: (a)Time-to-completion varying deadline constraints. (b)Images completed by each resource class. (c)The allocated EC2 nodes to achieve the deadline. (d)EC2 cost for renting instances.

References

1. Maintz, J.B.A., Viergever, M.A.: A survey of medical image registration. Medical Image Analysis **2**(1) (1998) 1–36
2. Hill, D.L.G., Batchelor, P.G., Holden, M., Hawkes, D.J.: Medical image registration. Physical Medical Biology **46** (1998) 1–45
3. Lester, H., Arridge, S.R.: A survey of hierarchical non-linear medical image registration. Pattern Recognition **32**(1) (1999) 129–149
4. Zitova, B., Flusser, J.: Image registration methods: A survey. Image Vision Computing **21**(11) (2003) 977–1000
5. Cena, B., Fox, N., Rees, J.: Fluid deformation of serial structural MRI for low-grade glioma growth analysis. In: Proc. International Conference on Medical Image Computing and Computer Assisted Intervention. Volume 3217., Rennes, France (2004) 1055–1063
6. Agostino, E., Maes, F., Vandermeulen, D., Suetens, P.: A viscous fluid model for multimodal non-rigid image registration using mutual information. Medical Image Analysis **7**(4) (2003) 565–575

7. Thirion, J.P.: Image matching as a diffusion process: An analogy with Maxwell demons. Medical Image Analysis **2**(3) (1998) 243–260
8. Vercauteren, T., Pennec, X., Perchant, A., Ayache, N.: Non-parametric diffeomorphic image registration with the demons algorithm. In: Proc. International Conference on Medical Image Computing and Computer Assisted Intervention. Volume 4792., Brisbane, Australia (2007) 319–326
9. Szeliski, R., Szeliski, R., Coughlan, J., Coughlan, J.: Hierarchical spline-based image registration. International Journal of Computer Vision (1994) 194–201
10. Rohlfing, T., Maurer, C.R., Bluemke, D.A., Jacobs, M.A.: Volume-preserving nonrigid registration of MR breast images using free-form deformation with an incompressibility constraint. **22** (2003) 730–741
11. Musse, O., Heitz, F., Armspach, J.P.: Topology preserving deformable image matching using constrainedhierarchical parametric models. IEEE Trans. Image Processing **10**(7) (2002) 1081–1093
12. Noblet, V., Heinrich, C., Heitz, F., Armspach, J.P.: 3-D deformable image registration: a topology preservation scheme based on hierarchical deformation models and interval analysis optimization. IEEE Trans. Image Processing **14**(5) (2005) 553–566
13. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Comm. of the ACM **24** (1981) 381–395
14. Chui, H., Rangarajan, A.: A new point matching algorithm for non-rigid registration. Computer Vision and Image Understanding **89**(2) (2003) 114–141
15. Kim, H., Parashar, M.: CometCloud: An Autonomic Cloud Engine. John Wiley and Sons, Inc., Hoboken, NJ, USA (2011)
16. Levi, K., Weiss, Y.: Learning object detection from a small number of examples: the importance of good features. In: Proc. IEEE International Conference on Computer Vision and Pattern Recognition. Volume 2., Washington, DC (2004) 53–60
17. Kim, H., Chaudhari, S., Parashar, M., Marty, C.: Online risk analytics on the cloud. In: Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on. (2009) 484–489