A Lightweight Execution Framework for Massive Independent Tasks

How to execute many independent tasks concurrently on distributed heterogeneous workstations?

Li Hui, Yu Huashan, and Li Xiaoming
Computer Networks and Distributed Systems Laboratory
Peking University
Website: http://net.pku.edu.cn
Outline

1 Motivation and problem definition
   • motivating applications
   • targeted execution environment
   • other possible solutions and the challenges

2 Our solution
   • the architecture of our solution
   • the optimizing strategies of our solution

3 Evaluations
   • comparison with GRAM
   • differences between Gracie and Falkon
Motivating applications

Alternative Splicing (AS): use mRNA and DNA to predict the possible protein templates.

ACTOR: remove haze and thin clouds from optical satellite imagery.

Characteristics of the applications:
1. Tasks are represented by prepared data, no interaction in the middle of execution
2. Hundreds of thousands tasks of the same type in a batch
3. They are independent
4. They are of unpredictable execution times with large variance
Targeted execution environment

1. Distributed workstations – possibly high latency
2. Managed – presumably reliable, e.g. computing facilities of different departments on a campus
3. Heterogeneous – different architectures, single core, multi-core, cluster (plan to), etc.
Other possible solutions based on Grids

- Grid middleware (GRAM Service, gLite).
- Local Resource Manager (PBS, Condor).

Works but not efficient enough

Users

GridWay

Globus/WS

SGE Cluster

PBS Cluster

Globus/WS

PBS Cluster

SGE Cluster

Globus/WS

PBS Cluster

SGE Cluster

Computer Networks and Distribute Systems Laboratory
Website: http://net.pku.edu.cn
The Limitations and Challenges

Limitations:
1. Use one Grid request for each task
2. Resource allocation strategy is request-oriented

A measurement of GRAM efficiency against manual operations

Challenges:
1. Low resource accessing efficiency
2. High overhead for task scheduling
Framework architecture of our solution

Be able to schedule multiple independent tasks in parallel on distributed machine resources.

The proposed framework consists of

- Grid APIs called in applications and communicate with resource manager
- Resource manager, monitoring computational nodes
- Adaptive job processor
Optimizing strategies of our solution

Three optimizing strategies have been devised to increase the execution efficiency and resource utilization:

1. **Task Grouping**: Pack up to thousands of tasks within one Grid request.
2. **User-oriented Resource Allocating**: Allocate resources to Grid application rather than any jobs or Grid requests.
3. **Balanced Task Workload Scheduling**: Balance task workload among multiple machine resources.

Computer Networks and Distributed Systems Laboratory
Website: http://net.pku.edu.cn
Task grouping strategy

Dynamically pack up to thousands of tasks within one Grid request.

Advantages:
1. To reduce the number of requests.
2. To utilize the parallel computing power of multiprocessor machines.

Disadvantages:
If the task execution lengths vary too much, there would be CPU resources wasted while waiting for the longer tasks to complete.

How to improve it?
User-Oriented resource allocating strategy

Allocate resources to application user rather than any Grid requests or jobs.

**Advantage:**
Reduce the costs for resources allocating, configuring and releasing.

10 short running tasks were executed on one SMP server with both the request-oriented strategy and request-oriented strategy. 22 seconds were saved by the optimizing strategy.
Balance task workload among machines with different computing power. Grid request is dynamically created, whose workload is calculated by an experience mathematic formula

\[ w_i = \partial \ast n_i \ast bw + \frac{(1 - \partial) \ast Tw \ast n_i}{2 \sum_{i=1}^{m} n_i} \]

**Advantage:**
Minimize the MTC’s time cost

Work time of 6SMPs for two strategies

Execute 51242 AS tasks on 6 SMPs in the PKU campus network with both balanced workload strategy and fixed workload strategy.
Implementation and application

The proposed framework and those strategies have been implemented in Gracie, a grid software platform developed by Peking University.

We have developed bioinformatics application AS and Blast with Gracie for China Bioinformatics Center.

Since Sept. 2007
More than 152,000 AS batch jobs have been processed
More than 246,000 AS results have been created
Comparison with GRAM

To compare Gracie with GRAM, we devised 4 policies to execute 2 kinds of AS task sets with different task granularity.

Four policies:
- GRAM, GRAM+Fork, Gracie, Gracie+Optimize.

Two kinds of task sets:
- CPU time less than 1 second.
- CPU time between 30 and 60s

Results showed Gracie uses less than 1/3 of time that GRAM needs to complete the two kinds of task sets.
THANK YOU ALL!

Computer Networks and Distributed Systems Laboratory
Website: http://net.pku.edu.cn
Differences between Gracie and Falkon

**Gracie**
Focus to schedule tasks on distributed machine resources with commodity processors and high-latency network.

**Falkon**
Focus to schedule tasks on high-performance cluster with low-latency network.

Distributed workstations/PCs on the internet

Cluster

Computer Networks and Distribute Systems Laboratory
Website: http://net.pku.edu.cn