Adaptive Grid Computation Approach in the Peer-to-Peer Grid Computing Systems

MaengSoon Baik, SungJin Choi, ChongSun Hwang, JunMin Gil and HeonChang Yoo

Dept. Computer Science & Engineering
Korea University and KISTI

Presented by: SungJin Choi
Adaptive Group Computation Approach in the Peer-to-Peer Grid Computing Systems

ABSTRACT

1. The peer-to-peer Grid computing systems has been regarded as an attractive platform to support massively parallel applications based on peer-to-peer computing technology in Grid computing area.
2. As the number of volunteers is increased, the job management overhead of the central job management server is more and more increased.
3. An adaptive group computation approach in the peer-to-peer Grid computing systems.
4. Reduce the job management overhead and the total computation time
5. KOREA@Home Project
OUTLINE

1. Introduction
2. Traditional Peer-to-Peer Grid Computing Systems Model
3. Adaptive Group Computation Approach
4. Implementation of Proposed Approach
5. Conclusion
1. Introduction

- Grid computing
  - Aims to offer pervasive access to a diverse collection of resources owned by different institutions through making virtual organization from resources in computation time.

- Peer-to-Peer Grid computing systems
  - Does not make any virtual organization in computation time
  - As the number of volunteers is increased, the job management overhead of the Central Parallel Job Management Server (CPJMS) is more and more increased.

- The architecture of the peer-to-peer Grid computing systems
  - Autonomous property of volunteers
  - Prevent volunteers from grouping according to their allocated job or registered resource semantics
  - Direct connection between the CPJMS and volunteers
  - High overhead
1. Introduction

- **Adaptive group computation approach**
  - Group coordinator => Intermediate Job Management Deputy (IJMD)
  - Computation members
  - Flowing volunteers

- **Computation groups**
  - IJMD
  - Computation members
  - Maintained by proposed algorithm

- **Contribution**
  - Reduces the overhead of the CPJMS
  - Shorten the total computation time using IJMD

- **KOREA@Home Project**
2. Traditional Peer-to-Peer Grid Computing Systems Model I

Parallel code and data distribution, maintenance of parallel job at execution and termination time in each volunteer

Final parallel execution result of each volunteer

Central Storage Server

Central Parallel Job Management Server

Related Applications
Adaptive Group Computation Approach in the Peer-to-Peer Grid Computing Systems

2. Traditional Peer-to-Peer Grid Computing Systems Model II

Peer-to-Peer Grid computing application

**Definition 1 (PPGCA)** Peer-to-peer Grid computing application allocated to a volunteer \( \text{vol}_i \), PPGCA\(_i\) is defined as followed.

\[
\begin{align*}
\text{PPGCA}_i &= p_{\text{code}}^{\text{vol}_i} + \bigoplus_{j=0}^{f_{\text{final}}^{\text{vol}_i}} D_{j}^{\text{vol}_i} \\
\bigoplus_{j=0}^{f_{\text{final}}^{\text{vol}_i}} D_{j}^{\text{vol}_i} &= D_{0}^{\text{vol}_i} \oplus \cdots \oplus D_{f_{\text{final}}^{\text{vol}_i}}^{\text{vol}_i}
\end{align*}
\]

\( p_{\text{code}}^{\text{vol}_i} \) is the parallel code allocated to volunteer \( \text{vol}_i \) developed by related application programmers, \( D_{j}^{\text{vol}_i} \) is \( j \)-th data unit allocated to volunteer \( \text{vol}_i \) by the CPJMS and \( \oplus \) is the sequential order having irreflexive, asymmetric and transitive relations.

Full data unit set

**Definition 2 (FDUS)** The full data unit set in peer-to-peer Grid computing is defined as follows.

\[
\begin{align*}
\text{FDUS} &= \bigcup_{i=0}^{\text{data}_{\text{unit}}^{\text{pool}}} D_{i} \\
\forall j, k \bigcup_{i=0}^{\text{data}_{\text{unit}}^{\text{pool}}} \geq \bigcup_{j=0}^{\text{vol}_f} [\bigoplus_{k=0}^{a_{\text{cf}}} D_{k}^{\text{vol}_j}]
\end{align*}
\]

\( \text{data}_{\text{unit}}^{\text{pool}} \) is the last number of data unit in data unit set to be executed, which submitted by related application programmers. In addition, \( \text{vol}_f \) is the last id number of registered volunteer allocated parallel job and \( a_{\text{cf}} \) is the last number of data unit allocated each volunteer.
Execution of volunteer

**Definition 3 (VE)** Execution of $i^{th}$ volunteer, $VE_i$, is defined as followed.

$$VE_i = \bigcup_{l=0}^{\alpha_{\text{order}}} \text{Ex}[p_{\text{code}}^{\text{vol}_i} \cup DU_l^{\text{vol}_i}]$$  \hspace{1cm} (3)

$DU_l^{\text{vol}_i}$ is $l$-th data unit allocated to $i$-th volunteer; $\text{Ex}[p_{\text{code}}^{\text{vol}_i} \cup DU_l^{\text{vol}_i}]$ is an execution by $i$-th volunteer using parallel code $p_{\text{code}}$ and allocated data unit $DU_l$, and $\bigcup_{l=0}^{\alpha_{\text{order}}}$ means a sequential execution order having irreflexive, asymmetric and transitive relation to the last allocation data unit number, $\alpha_{\text{order}}$.

Property of parallel code and data unit

**Property 1 (PCIP)** Parallel codes and dataset allocated to each volunteer satisfy followed conditions.

- $\forall i \neq j \quad p_{\text{code}}^{\text{vol}_i} = p_{\text{code}}^{\text{vol}_j}$
- $\forall i \neq j, l \neq k \quad DU_l^{\text{vol}_i} \neq DU_k^{\text{vol}_j}$  \hspace{1cm} (4)
Adaptive Group Computation Approach in the Peer-to-Peer Grid Computing Systems

3. Adaptive Group Computation Approach I

- Application program submission
  - Parallel code and full data unit set
  - The average computation time list per resource

Table 1. An example of the average computation time list per resource

<table>
<thead>
<tr>
<th>OS</th>
<th>CPU</th>
<th>Memory</th>
<th>Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2(S)</td>
<td>Pen IV 2.4</td>
<td>512</td>
<td>0.8 minutes</td>
</tr>
<tr>
<td>W2(P)</td>
<td>Pen III 750M</td>
<td>256</td>
<td>2 minutes</td>
</tr>
<tr>
<td>W 98</td>
<td>Celeron 1.7G</td>
<td>256</td>
<td>1.8 minutes</td>
</tr>
<tr>
<td>W 98</td>
<td>Pen II 450M</td>
<td>128</td>
<td>3 minutes</td>
</tr>
</tbody>
</table>
3. Adaptive Group Computation Approach II

- **Volunteer registration**
  - Local operating systems type
  - CPU type
  - Memory capacity
  - Usable local hard disk capacity
  - Volunteer address type
  - Using network bandwidth type
  - Resource submitting state
    - Time reserved: stably providing the resources during the reserved time
    - Only registered: not-stably providing the resources
  - Registered volunteer table
3. Adaptive Group Computation Approach III

Making computation group members I

Selection procedure

- Latest state message: to update the registered volunteer table
- Required IJMD

**Condition 1 (RIJMD)** Volunteer, \( \text{vol}_i \), satisfied followed conditions becomes a candidate of IJMD.

\[
\begin{align*}
ULHD C(\text{vol}_i) & \geq \Psi_{\text{member}} \Pi_{DU} Avg_{DU}^{size} \\
\text{ReservedTime}(\text{vol}_i) & \geq \xi_{time} + \Omega
\end{align*}
\]


\( ULHD C(\text{vol}_i) \) is an usable local hard disk capacity registered by volunteer \( \text{vol}_i \), \( \Psi_{\text{member}} \) is the number of computation group members determined by systems, \( \Pi_{DU} \) is the number of data unit possessed at IJMD determined by systems for data unit distribution to computation group members, \( Avg_{DU}^{size} \) is an average data unit size in full data unit set, \( \text{ReservedTime}(\text{vol}_i) \) is the reserved time registered by volunteer \( \text{vol}_i \) in registered volunteer table, \( Avg(\xi_{time}) \) is an average computation time per all resources acquired from the average computation time list, \( \xi_{time} \) is the required alive time for IJMD determined by systems, \( \Omega \) is a surplus for network delay between an IJMD and computation group members.

Reserving ready message

Reserving message
3. Adaptive Group Computation Approach IV

Making computation group members II

- Matching computation group

- Computation group member list table: CPJMS -> IJMD, Members
- Matchmaking message: IJMD -> Members
- Acknowledge for matchmaking message: Members -> IJMD
- NOK message: Members -> IJMD if not received the matchmaking msg.
- Network failure occurrence message: Members -> CPJMS
- Remove message: CPJMS -> IJMD
Making computation group members III

Computation group matchmaking algorithm

After selecting an IJMD and computation group members
Do
The CPJMS: make(computation group members list table);
The CPJMS: send(computation group members list table) to selected IJMD;
The CPJMS: send(IP address of the selected IJMD) to all computation group members in the computation group members list table;
The selected IJMD: if receive (the computation group members list table) from the CPJMS;
The selected IJMD: then send (the matchmaking message) to all computation group members in the computation group members list table;
The selected IJMD: if receive (the acknowledge for matchmaking message) from all computation group members in the computation group members list table;
The selected IJMD: then matchmaking is finished;
The selected IJMD: exit;
The selected IJMD: f;
The selected IJMD: else if receive (the NOK message) from volunteer, vol_6;
The selected IJMD: then resend (the matchmaking message) to volunteer, vol_6;
The selected IJMD: if receive (the acknowledge for matchmaking message) from all computation group members in the computation group members list table;
The selected IJMD: then matchmaking is finished;
The selected IJMD: exit;
The selected IJMD: f;
The selected IJMD: f;
The selected IJMD: else if receive (the remove message) from the CPJMS or (the network failure occurrence message) from volunteer, vol_6;
The selected IJMD: then remove (computation group members piggybacked in message) or (computation group members, vol_6) from the computation group members list table;
The selected IJMD: if receive (the acknowledge for matchmaking message) from all computation group members list table, except volunteers piggybacked the remove message and vol_6;
The selected IJMD: then matchmaking is finished;
The selected IJMD: exit;
The selected IJMD: f;
The selected IJMD: f;
The selected IJMD: else;
The selected IJMD: matchmaking is failed;
The selected IJMD: f;
3. Adaptive Group Computation Approach VI

- Parallel code and data unit distribution
- Parallel job matching table (data unit distribution table)

Table 2. An example of parallel job matching table between volunteer and allocated parallel job

<table>
<thead>
<tr>
<th>Volunteer ID</th>
<th>Parallel Job ID</th>
<th>Allocating Data Set</th>
<th>Current Allocated Data Set</th>
<th>Volunteer State</th>
</tr>
</thead>
<tbody>
<tr>
<td>vol_0</td>
<td>PJD_0</td>
<td>( \sum_{i=0}^{D(PJD_0)} DU_{i}^{vol_0} )</td>
<td>( \sum_{i=0}^{C(PJD_0)} DU_{i}^{vol_0} )</td>
<td>Executing</td>
</tr>
<tr>
<td>vol_1</td>
<td>PJD_1</td>
<td>( \sum_{i=0}^{D(PJD_1)} DU_{i}^{vol_1} )</td>
<td>( \sum_{i=0}^{C(PJD_1)} DU_{i}^{vol_1} )</td>
<td>Executing</td>
</tr>
<tr>
<td>vol_2</td>
<td>PJD_2</td>
<td>( \sum_{i=0}^{D(PJD_2)} DU_{i}^{vol_2} )</td>
<td>( \sum_{i=0}^{C(PJD_2)} DU_{i}^{vol_2} )</td>
<td>Removed</td>
</tr>
<tr>
<td>vol_3</td>
<td>PJD_3</td>
<td>( \sum_{i=0}^{D(PJD_3)} DU_{i}^{vol_3} )</td>
<td>( \sum_{i=0}^{C(PJD_3)} DU_{i}^{vol_3} )</td>
<td>Executing</td>
</tr>
<tr>
<td>vol_4</td>
<td>PJD_4</td>
<td>( \sum_{i=0}^{D(PJD_4)} DU_{i}^{vol_4} )</td>
<td>( \sum_{i=0}^{C(PJD_4)} DU_{i}^{vol_4} )</td>
<td>Executing</td>
</tr>
<tr>
<td>vol_5</td>
<td>PJD_5</td>
<td>( \sum_{i=0}^{D(PJD_5)} DU_{i}^{vol_5} )</td>
<td>( \sum_{i=0}^{C(PJD_5)} DU_{i}^{vol_5} )</td>
<td>Removed</td>
</tr>
</tbody>
</table>
3. Adaptive Group Computation Approach VII

- Maintenance mechanism for computation group
  - Failure of IJMD (the CPJMS detecting)
    - Deputy failure message by the CPJMS: CPJMS -> Members
    - Failure event confirm message: Members -> CPJMS
  - Failure of IJMD (computation member detecting)
    - Deputy failure message by computation members: Members -> CPJMS
    - Failure confirm message: CPJMS -> IJMD
    - Network failure message: CPJMS -> IJMD
  - Failure of computation members
    - The member failure message by the IJMD: IJMD -> CPJMS
Correctness I

**Lemma 1** If a failure is occurred in the IJMD, then the CPJMS detects an occurrence of a failure.

**Proof:** Assume that a failure is occurred in the IJMD, then the IJMD cannot sends the alive message to the CPJMS until time out interval and the CPJMS detects an occurrence of a failure in the IJMD. Although the network failure is occurred between the IJMD and the CPJMS, an occurrence of a failure in the IJMD is detected by computation group members because they periodically sends the alive message and request more data unit. If computation group members does not receive acknowledgement for alive message or requested more data unit, it sends the deputy failure message by computation members to the CPJMS. Therefore, if a failure is occurred in the IJMD, the CPJMS detects an occurrence of a failure.

**Lemma 2** If a failure is occurred in any computation group member, the IJMD and the CPJMS detects a failure.

**Proof:** Assume that a failure is occurred in any computation member, then the computation group member does not send the alive message to the IJMD and the IJMD detects an occurrence of a failure in the computation group member. The IJMD detects an occurrence of a failure sends the member failure message by the IJMD to the CPJMS. Therefore, if a failure is occurred in any computation group member, the IJMD and the CPJMS detects a failure.
3. Adaptive Group Computation Approach IX

Correctness II

**Lemma 3** Although a failure is occurred in the IJMD, the alive computation group members continue their group computation.

**Proof:** Assume that a failure is occurred in the IJMD, then the CPJMS confirms the aliveness of computation group members and selects the new IJMD from registered volunteer table. The newly selected IJMD takes the matchmaking procedure with alive computation group members. When the matchmaking procedure is successfully finished, the CPJMS sends the data unit distribution table to the newly selected IJMD. Therefore, although a failure is occurred in the IJMD, alive computation group members continue group computation with the newly selected IJMD.

**Lemma 4** Although a failure is occurred in any computation group member, the IJMD continues the group computation.

**Proof:** Assume that a failure is occurred in the computation group member, then the IJMD and the CPJMS detect an occurrence of a failure in the computation group member and remove this computation group member from the computation group member list table and the data unit distribution table. Therefore, although a failure is occurred in any computation group member, the IJMD continues the group computation.

**Theorem 1** The group computation is continued in spite of an occurrence of a failure in the IJMD or any computation member.

**Proof:** The proof of this theorem is through by above four lemmas. That is, the CPJMS detects a failure in the IJMD or computation group members. The CPJMS continues the group computation through selecting new IJMD in IJMD failure case or remove computation group members from the computation group member list table and the data unit distribution table in computation group member failure case.
The KOREA@Home Project

Korea@Home

Introduction
About Korea@Home / Contact Us
Korea@Home Platform / Applications
Statistics
Status / Performance

Korea@Home is “New Future of Korea!”

Target Applications
Since 2002, Korea@Home has tried to search target applications for internet-based distributed computing in various fields. The internet-based distributed computing is suitable for the large-scale applications, in which each job is small enough to be calculated in each volunteer’s PC and all jobs are not dependent on each other. Currently, Korea@Home system is performing two kinds of target applications.

Korea@Home Statistics
People can participate in Korea@Home from anywhere on Internet to contribute their own PC’s idle time. Now, Korea@Home has 1,086 Volunteers and 4,025 Agents, and its performance is 694 GFlops in maximum and 770 GFlops in average for best 24 hours.
Adaptive Group Computation Approach in the Peer-to-Peer Grid Computing Systems

4. Implementation of Proposed Approach II

The KOREA@Home Project II

(a) Volunteer execution screen in KOREA@Home Project

(b) Distribution chart by CPU type of volunteers in KOREA@Home Project
Current implementation of proposed approach I
(according to the number of computation group members)
Current implementation of proposed approach II (according to the registered time of the IJMD)
5. Conclusion

- Adaptive grid computation approach in the peer-to-peer Grid computing systems
- Computation group maintenance mechanism and correctness
- KOREA@Home Project (http://www.koreaathome.org/eng/)
- Experiment result of proposed approach