Resource Allocation in Distributed Systems based on Incomplete Information

Jörg Schneider
Advance Reservation
Coupling Resources in the Grid
Coupling Resources in the Grid
Coupling Resources in the Grid
Coupling Resources in the Grid

- Only advance reservations
- Independent resources
  - Multi-organizational
  - Local resource management system with SLAs
  - Local user
  - Information hiding
Resource Model

- **homogeneous**: same duration on all resources
- **heterogeneous**: duration depends on speed
- **moldable jobs**: duration depends on capacity
Grid Workflow
Grid Workflow (co-allocation)
Questions

- How to define Grid Workflows?
- Fast Answer: Can we accept the SLA?
- Optimizing the Schedule
- Repairing the schedule
Questions

- How to define Grid Workflows?
- Fast Answer: Can we accept the SLA?
- Optimizing the Schedule
- Repairing the schedule
Grid Workflow Language

Grid workflow language
- Mandatory: Task description (executable)
- Optional:
  - Dependencies
  - Workload
  - resources to be used
  - Communication
  - SLA / QoS properties
- multiple representations

Representation
- (formal) textual language (script, XML)
- (formal) visual language (DAG, petri net)
- data structure definition
Questions

- How to define Grid Workflows?
- Fast Answer: Can we accept the SLA?
- Optimizing the Schedule
- Repairing the schedule
Fast Admission of Grid Workflows

- **Answer to Request**
  - Yes: There is definitely enough capacity
  - No: There may be not enough capacity

- **Negotiate mapping and start time during admission**

- **Reservation protocol (broker ↔ resource)**
  - **Probe:** Ask for reservation candidates
  - **Reserve:** Preliminary reserve
  - **Commit/Cancel**
Algorithms

- **Greedy:**
  - Simple „schedule next ready job“
  - HEFT - Heterogeneous Earliest-Finish-Time
    - HEFTSync – handles co-allocations
    - HLST – starts at the end

- **Optimization**
  - Constraint solver
  - Genetic algorithm
Questions

• How to define Grid Workflows?

• Fast Answer: Can we accept the SLA?

• Optimizing the Schedule

• Repairing the schedule
Measuring Fragmentation

- Count free blocks

- Largest block

- Average with preference for large blocks:

\[
F = 1 - \frac{\sum_{i=1}^{n} f_i^2}{\left(\sum_{i=1}^{n} f_i\right)^2}
\]
Optimizing the Schedule

- **Local only**
  - Only within resource schedule
  - Timing of Grid jobs fix

- **Local initiated**
  - Only within resource schedule
  - Request alternate timing for Grid jobs

- **Grid initiated**
  - Grid and resource schedules
  - Request alternate timing from resources
  - Shadow schedule to try out combinations
Questions

Spec
- How to define Grid Workflows?

Online
- Fast Answer: Can we accept the SLA?

Offline
- Optimizing the Schedule

Rescue
- Repairing the schedule
Resource break down

- Recovers after unpredictably long time
- No notification in advance
- Affected jobs:
  - Currently running (not handled)
  - Planned during break down (will be remapped)
- No postponing or restarting

Rescue goal:

- Save as much workflows (workload) as possible
Recovery process
Recovery process

rescue horizon

downtime

resource

t₀

time

j₅

j₁₀

j₆

j₇
Recovery process

rescue horizon

downtime

resource

t₀

time

j₁₁

j₁₀

j₁₂

j₅

j₆

j₇
Recovery process
Model

- **Determine rescue interval**
  - Critical time = threshold for reservation success
  - Based on average load + predicted submissions
  - Feedback based (try till successful)

- **Remapping**
  - Resubmit affected job (fast)
  - Resubmit affected workflow
  - Use optimization strategy
Questions

- **Spec**
  - How to define Grid Workflows?

- **Online**
  - Fast Answer: Can we accept the SLA?

- **Offline**
  - Optimizing the Schedule

- **Rescue**
  - Repairing the schedule
Taxonomy

- **Base Concept**
  - *control flow, data flow, state transitions, rule based*

- **Resource Allocation**
  - Workflow (*concrete/abstract*)
  - job description (*commandline, service, dictionary*)
  - duration (*N/A, fixed time, speed-up model*)
  - data transfer (*N/A, with or without amount*)
  - shared data (*N/A, repository*)
  - resource requirements (*dictionary, specific selection*)
Taxonomy II

- **Control structure**
  - sequential execution
  - parallel execution
  - conditional execution *(return code, complex conditions)*
  - loops *(fixed iterations, conditional)*
  - interactivity
  - co-allocation
  - primitive operations

- **Data transfer**
  - interjob communication ()
  - intrajob communication
  - files
  - concurrent access
Quality of Service
- time (deadline, earliest start time)
- level (task, workflow)
- Exceptions (handle unpredicted behavior)
- rescue workflow (predicted failures)
- additional QoS (security, availability)
- Service Level Agreement

Semantic (formal, informal)
- behavior
- task description
- task requirements
HEFT: Rejection Ratio and Utilization

![Graph showing rejection ratio and average utilization against normalized mean inter arrival time.](image)
HEFT: Average Scheduling Duration

![Graph showing the average scheduling duration for HEFTSync and HEFTSyncBT over normalized mean inter arrival time.](image)
HEFT: Run Time / Workflow Size

Experiment Duration (HEFTSyncBT)
Experiment Duration (HEFTSync)

Time in Seconds

Max. Number of Activities
Fragmentation vs. rejection
Rescue: Load-Based Remapping

- Remapping Interval: $i_\eta(t_0)$
  - For each $t > i_\eta(t_0)$ must be $\tilde{l}_{t_0}(t) < \eta$
Rescue: Termination Ratio

\[ \text{load}_{\text{global}} > \text{load}_{\text{local}} \]

\[ \text{load}_{\text{global}} < \text{load}_{\text{local}} \]
Rescue: Remapping Overhead

\[ \text{load}_{\text{global}} \gg \text{load}_{\text{local}} \]

\[ \text{load}_{\text{global}} \ll \text{load}_{\text{local}} \]

- Remapping overhead (%) vs. failure duration
- Central vs. Distributed

Graphs showing remapping overhead for different load conditions.