Elastic Grid Reservations with User-Defined Optimization Policies

Workshop on Adaptive Grid Middleware, 30th September 2004

Thomas Röblitz, Florian Schintke and Jan Wendler
{roeblitz,schintke,wendler}@zib.de
Zuse Institute Berlin
Why do we need reservations?

*Higher guarantee that specific resource allocations succeed!*

- Hotel reservation, ticket reservation, ...
- CPU reservation, bandwidth reservation, ...
- Reservations have been already studied in many areas incl. network management, CPU cycle scheduling and Grid computing.

> What’s still to be done? *Flexibility & Efficiency*
How do we reserve? – Rigid Parameters

Rigid request: Reserve 16 CPUs at cluster ELFIE from 30/09/04 10:00 til 30/09/04 12:00.

Alternatives:
- location (cluster)
- start time
- number of CPUs, duration

User jobs
- user A
- user B
- user C

User reserv.
- user D

Reserv. Req.
- request

Alternatives:
- location (cluster)
- start time
- number of CPUs, duration

Time轴: 30/09/04 06:00 - 12:00

Jobs:
- job 39405
- job 39406
- job 39408
- job 39409
- job 39410
- job 39411
- job 39412
- job 39413
- job 39414
- job 39415

Conflicts:
- resv D.0

Nodes: c0-0, c0-1, c0-2, ..., c0-15
Elastic request: Reserve 12-16 CPUs at cluster ELFIE for 2 hours from 30/09/04 10:00 til 30/09/04 13:00 with s=0.1, p=0.9 (speedup parameters*).

Determine alternative duration:
- Assume 2 hours given for 16 CPUs ($d_{\text{ref}}=2$, $N_{\text{ref}}=16$).
- $\text{dur}(N) = \text{sp}(N_{\text{ref}}) \times d_{\text{ref}} / \text{sp}(N)$
- $\text{dur}(15) = 02:03\,\text{h}$, $\text{dur}(14) = 02:06\,\text{h}$,
- $\text{dur}(13) = 02:10\,\text{h}$, $\text{dur}(12) = 02:14\,\text{h}$

*Speedup sp defined as $\text{sp}(N) = 1 / (s + p/N)$;
  s - sequential part, p – parallel part.
  (G. Amdahl 1967, Validity of the single-processor approach to achieving large scale computing capabilities)
How do we reserve? – All together

Flexible aspects of a reservation request:

- CPU range
- time range
- location

Elastic request: Reserve 8-16 CPUs at ANY cluster for 2 hours from 30/09/04 10:00 til 30/09/04 13:00 with s=0.1, p=0.9 (speedup parameters). The duration is given for 16 P4 (2GHz) CPUs.

Attributes of a request:

- CPU range: \( np_{\text{min}}, np_{\text{max}} \)
- time range: earliest start time (est), duration (dur), latest end time (let)
- CPU-time relation: speedup (sp)
- performance: \( np_{\text{ref}}, dur_{\text{ref}}, perf_{\text{ref}} \)
- miscellaneous, e.g. uid/gid/certificate
Algorithm – Goals

• **efficiency** – number of reservation tries in case of success/failure
  ➔ tradeoff between flexibility and reservation tries

• **impact on user jobs** – longer wait time
  ➔ reservations reduce the optimization space for a local scheduler
Algorithm – Services & Interactions

1. GIS Phase
2. Probe Phase
3. Reserve Phase
Algorithm – Probe Phase Overview

- **goal:** minimize number of tries in case of success \textbf{AND} failure
- **idea:**
  - obtain \textbf{additional information} about system utilization from the CRSs
  - only consider candidates where the system utilization promises success
    \( \Rightarrow \) reservation probability value (esr – estimated success rate)

\textbf{Note!} Mechanism can be used for arbitrary attributes (e.g. cost).
Algorithm – Probe Phase ESR

- Domain for esr is \([0,1] \subset R\).
- **static**: the longer the book-ahead period \(t\) is the higher is the esr value
  
  \[ esr(t,h) := 1 - e^{-\frac{t}{h}} \]

- **history**: determine average of idle processors \(idle^{NP}\) for the requested time range using a `calendar` of logged utilization records

  \[ esr(t,np) := \begin{cases} 
  1 & , 2np < idle^{NP}(t) \\
  2 + \frac{2np}{idle^{NP}(t)} & , idle^{NP} \leq \frac{2}{n} \leq idle^{NP}(t) \\
  0 & , \text{elsewise} 
  \end{cases} \]

- **load**: approximate the time \(T_{wkl}\) that is reached, when the current workload (running + idle jobs and existing reservations) is finished

  \[ esr(t,T_{wkl}) := \begin{cases} 
  1 & , t \geq T_{wkl} \\
  0 & , t < T_{wkl} 
  \end{cases} \]
Algorithm – Reserve Phase Overview

- **problem:** many candidates
- **goal:** select `best` candidate among the many
- **idea:** sort the possible candidates according to the user's preferences
- **preferences are prioritized,** for example
  - 1st level: *end time*
  - 2nd level: *np* (number of processors)
  - 3rd level: *cost*
- Any metric may be used within the preferences

⇒ GRS probes the CRSs to calculate the values of the metrics that are used in the preferences
Algorithm – Reserve Phase Preferences Mgmt.

• **idea:** construct tree where the $i$-th level is sorted according to the $i$-th preference

• **example:** consider the following candidates

\{(site, start, end, np, costs)\} = \{(A, 0, 4, 8, 10), (A, 1, 4, 16, 15), (A, 2, 6, 8, 9), (A, 3, 6, 16, 13), (B, 1, 4, 8, 12), (B, 2, 4, 16, 16), (B, 3, 6, 8, 8), (B, 4, 6, 16, 8)\}

1st level: **end time**

2nd level: **np** (number of processors)

3rd level: **costs** (costs of the reservation)

• traverse tree in **depth-first** manner
Evaluation

• do not use/simulate the Grid Information Service (GIS) (*only one cluster*)

• use a workload trace from a real system and simulate it with the Maui scheduler
  - subset (2k jobs) of the workload for the SDSC IBM Blue Horizon (144 8-way nodes)

• generate a reservation workload based on the subset of jobs
  - important parameters are book-ahead time, time range, processor range
Evaluation – Selected Results

1. **efficiency**: 1st load, 2nd static, 3rd history
   - the more filtered out by threshold, the less tries

2. **impact on user jobs:**
   - varies wrt. book-ahead time
   - few jobs wait **shorter**, more jobs wait **longer**
Evaluation - Efficiency

• main criteria: number of reservation tries (reserve phase) in case of success and failure
• GRS filters candidates with a low esr value (less than a threshold)
• remaining are sorted according to user preferences
Evaluation - Efficiency

• main criteria: number of reservation tries (reserve phase) in case of success and failure

- GRS filters candidates with a low esr value (less than a threshold)
- Remaining are sorted according to user preferences

---

Bar chart: average for successful/failed reservation requests

- esr calculation method & requests result
- candidates after probe request
- candidates after esr filter
- rejected by local scheduler
- rejected by CRS check
Evaluation – Impact on User Jobs

- reservations reduce optimization possibilities for a local scheduler
- resources may stay idle (before reservations begin)

⇒ some jobs wait longer (most likely)
Evaluation – Impact on User Jobs

- Reservations reduce optimization possibilities for a local scheduler.
- Resources may stay idle (before reservations begin).
- Some jobs wait longer (most likely).
• reservations reduce optimization possibilities for a local scheduler
• resources may stay idle (before reservations begin)
  some jobs wait longer (most likely)
Conclusion

• Reservation requests should be **elastic** to allow optimizations (site, start, #CPUs).

• Processing elastic reservation requests requires:
  - efficient algorithms
  - possibility to define user policies

  ➔ Introduced a **probe** phase for calculating properties of reservation candidates:
  - extensible,
  - efficient &
  - user preferences
Thanks!

Questions?

Coffee Break!