Planning and Optimization in TORQUE Resource Manager

Dalibor Klusáček\textsuperscript{1,2}  
Václav Chlumský\textsuperscript{1}  
Hana Rudová\textsuperscript{2}

\textsuperscript{1}CESNET, Czech Republic  
\textsuperscript{2}Faculty of Informatics, Masaryk University, Czech Republic

klusacek@cesnet.cz

HPDC 2015, Portland, Oregon, USA
Overview

• Contribution
  – new scheduler for TORQUE RM
  – job schedule optimized by a metaheuristic
  – improves the quality of job schedule (initial schedule built by backfilling)
  – applied in practice (CERIT-SC system, ~ 5,000 CPUs, 7 clusters)

• State of the art
  – queue-based schedulers (PBS, Moab, Maui, Slurm, …)
  – backfilling (optimizes resource utilization/wait time/slowdown)
  – further “tailoring” (fair-share, priorities, per user/group limits, …)
Importance

- **Metaheuristics are popular in “theoretical” works**
  - results indicate improved performance wrt. current solutions
  - actual implementations and applications are very rare

- **It is quite hard to make it work in the real life...**
  - “theoretical” models are far from the needs of real providers/users
    - fast decisions
    - detailed system setups (priority, limits, fairness, …)
    - multi-criteria optimization problems (performance, fairness, …)
    - complex job specifications, job dependencies, SW licenses …
Applied Solution

- Initial schedule built by **conservative backfilling**
- Schedule is periodically optimized using a **local-search inspired metaheuristic**, optimizing
  - performance (wait time and slowdown)
  - fairness (fair-share-like “max-min” approach)

**Schedule evaluation**
- wait time
- slowdown
- fairness

**Schedule optimization**
- local search-inspired metaheuristic
- random job re-allocations

**accept / reject** modifications
Realism

• All major features of “classic” schedulers are supported
  – adaptation to dynamic events (inaccurate estimates, failures)
  – support of various limits concerning max. exec. time/CPU
  – complex job specifications (CPUs, RAM, HDD, SW-licenses,...)
  – multi-resource fair-sharing (CPU and RAM consumption)
  – inter-job dependencies
  – maintenance-aware planning (assuring that jobs complete prior a maintenance period)
Deployment

- **CERIT-SC system** (~ 5,000 CPUs, 7 clusters)
- Since July 2014 (11 months)
- “before – after” comparison

![Graphs showing comparison between old and new schedulers for system utilization, wait time, and slowdown.](image)
Conclusion

- **Realistic application of a metaheuristic**
  - improved performance both in the simulations and in the reality
  - detects and fixes “pathological” job assignments
- **Schedule (execution plan)** is available to the users and system administrators
  - (partial) predictability (planned start times may change)
- **Easy (advanced) problem detection**
  - bad job specification (no planned start time, very frequent)
- **Easier setup of critical system constraints**
  - e.g., too strict resource limits (planned start times are very high)