Distributed QoS-Aware Scheduling in Storm

In the 9th ACM International Conference on Distributed Event-Based Systems

Outline

- Qos-aware scheduling
- Implement distributed scheduling in Storm
- Experiment
- Conclution

Qos-aware scheduling

- Goal:
  - Implement a distributed scheduling algorithm that is aware of QoS attributes - **latency, availability and utilization**.

- Cost Space:
  - 4-dimension metric space including :
    - 2-dimension refer to latency
    - 2-dimension refer to availability and utilization

Qos-aware scheduling

- Placement Algorithm including:
  - Virtual Placement Algorithm
  - Physical Placement Algorithm

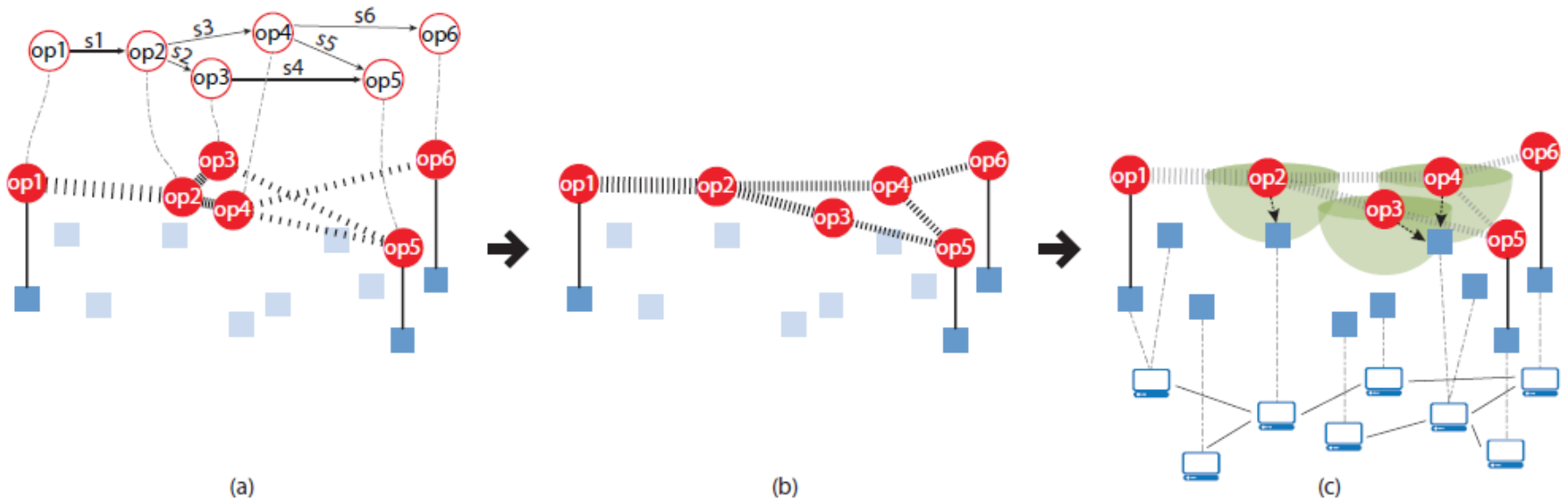


Figure 1: High-level architecture of our solution: (a) pinned and unpinned operators; (b) Virtual Operator Placement; (c) Physical Operator Placement

Qos-aware scheduling

- Virtual Placement Algorithm
  - Solve **the minimum network usage** configuration of the operators is like to solve **the minimum energy** configuration of th**e spring system**
  - each operator **op*i*** moves = the force **F*i***
  - the latency **Lat(*l*)** = spring extension **s*i***
  - the data rate over that link **DR(*l*)** = the spring constant **k*l***

  - **=> op*i*** moves = **DR(*l*)** x **Lat(*l*)**

Qos-aware scheduling

- Physical Placement Algorithm
- The distance between P*i* = (Ppl1*i*, Ppl2*i*, Pa*i*, Pu*i*) and P*j* = (Ppl1*j*, Ppl2*j*, Pa*j*, Pu*j*) is computed as follows:

$$d(\vec{P}_i, \vec{P}_j) = \sqrt{\begin{array}{ll} w_l^2[(\frac{P_{l1i}-P_{l1j}}{Lat_{max}})^2 & +(\frac{P_{l2i}-P_{l2j}}{Lat_{max}})^2]+ \\ w_a^2(P_{ai}-P_{aj})^2 & +w_u^2(P_{ui}-P_{uj})^2. \end{array}}$$

Implement distributed scheduling in Storm

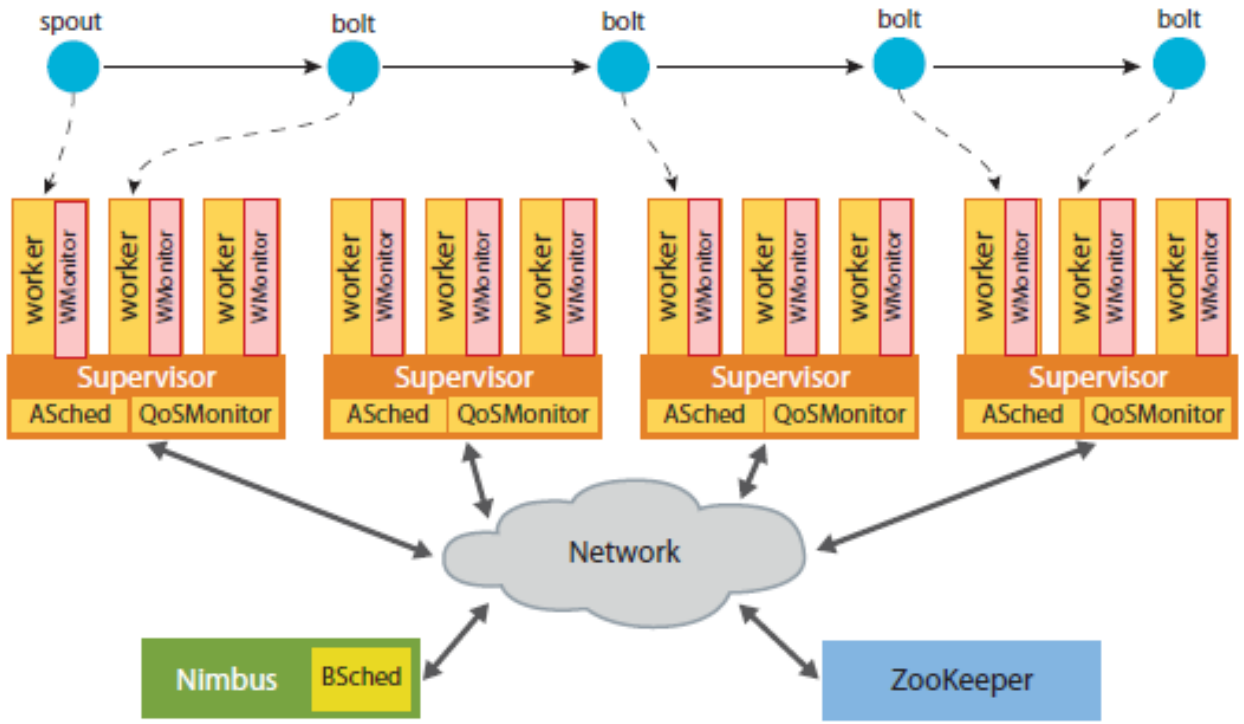


Figure 2: Extended Storm architecture: AdaptiveScheduler is abbreviated as ASched, WorkerMonitor as WMonitor, and BootstrapScheduler as BSched

Implement distributed scheduling in Storm

- QoSMonitor: provides the QoS awareness to each distributed scheduler
- AdaptiveScheduler: executes the distributed scheduling policy on every worker node.
- A single loop iteration is executed periodically (every 30 s), and is composed by the following phases of the MAPE reference model for autonomic systems:
  - **Monitor, Analyze, Plan, and Execute**

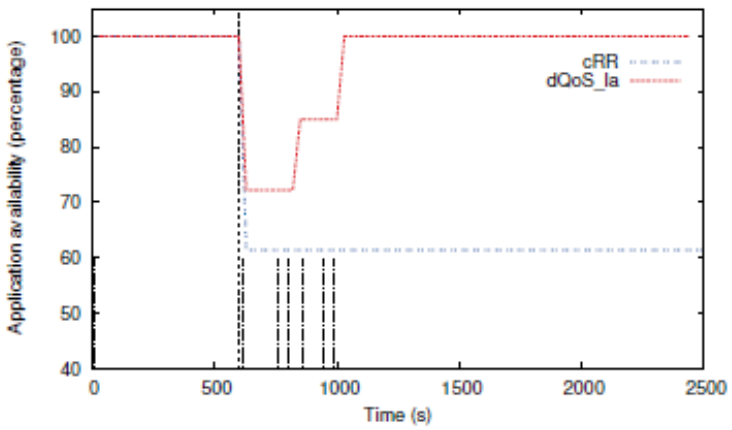
Implement distributed scheduling in Storm

- Monitor phase:
  - the AdaptiveScheduler acquires the information collected by the QoSMonitor and identifies the set of local executors that could be moved.
- Analysis phase
  - the AdaptiveScheduler runs the Virtual Placement Algorithm
- Plane:
  - determine which worker node will execute e. To this end, the planner executes the Physical Placement Algorithm to find the worker node closest to ~P which has at least a free worker slot and can thus host e

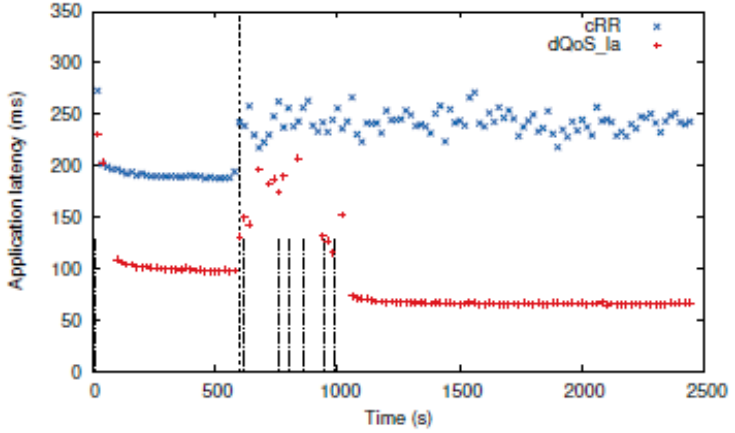
Implement distributed scheduling in Storm

- Excute phase:
  - if a new assignment must take place, the executor eis moved to the new candidate node.
  - The new assignment decision is shared with the involved worker nodes through ZooKeeper

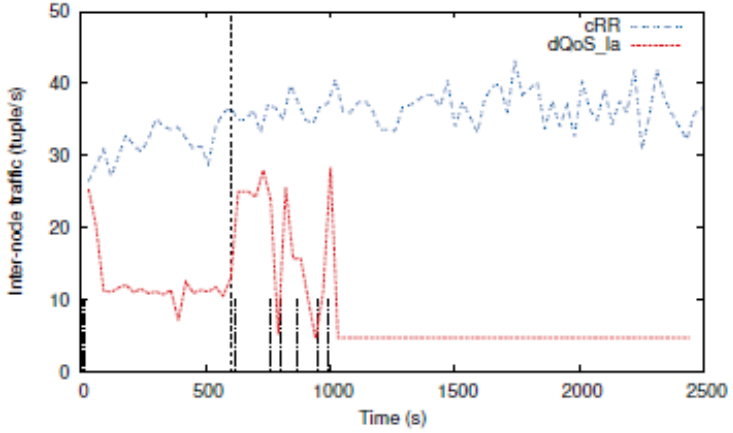
Experiment 1



(a) Application availability

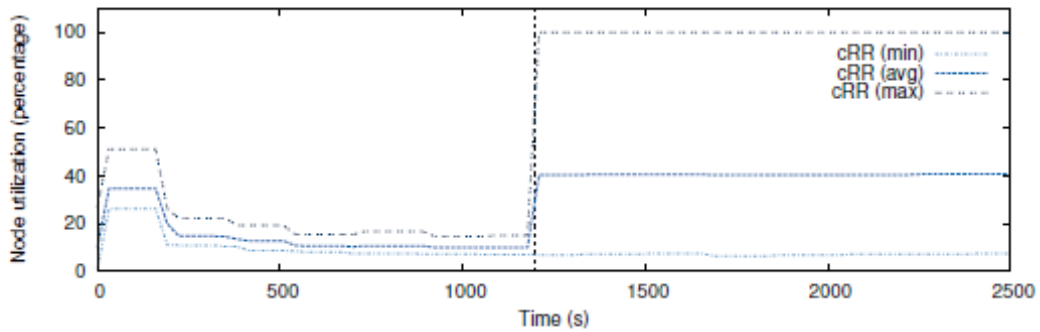


(b) Application latency

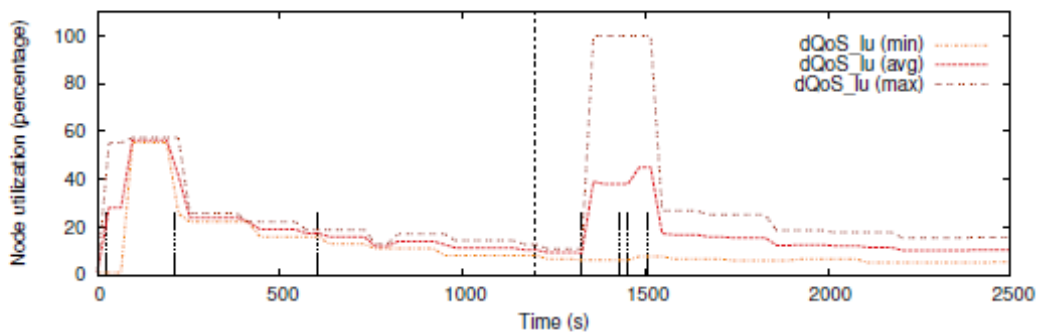


(c) Inter-node traffic

Experiment 2



(a) Node utilization for cRR



(b) Node utilization for dQoS_lu

CONCLUSIONS

- designed and implemented a distributed QoS-aware scheduler for DSP systems based on Storm.
- outperforms than centralized default one
- enhances the system with adaptation capabilities to react to changes in a distributed fashion.