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
• Conclusion
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

- Place the article

   • Placement Algorithm including:
     • Virtual Placement Algorithm
     • Physical Placement Algorithm

![Diagram showing different placement algorithms](image)

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 the spring system
  • each operator $\text{opi}$ moves = the force $F_i$
  • the latency $\text{Lat}(l) = \text{spring extension } s_i$
  • the data rate over that link $\text{DR}(l) = \text{the spring constant } k_l$

• $\Rightarrow \text{opi moves} = \text{DR}(l) \times \text{Lat}(l)$
Qos-aware scheduling

- Physical Placement Algorithm
- The distance between $P_i = (P_{pl1i}, P_{pl2i}, P_{ai}, P_{ui})$ and $P_j = (P_{pl1j}, P_{pl2j}, P_{aj}, P_{uj})$ is computed as follows:

$$d(\vec{P}_i, \vec{P}_j) = \sqrt{w_l^2\left[\left(\frac{P_{l1i} - P_{l1j}}{Lat_{max}}\right)^2 + \left(\frac{P_{l2i} - P_{l2j}}{Lat_{max}}\right)^2\right] + w_a^2(P_{ai} - P_{aj})^2 + w_u^2(P_{ui} - P_{uj})^2}.$$
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 \( \sim P \) which has at least a free worker slot and can thus host e
Implement distributed scheduling in Storm

• Execute phase:
  • if a new assignment must take place, the executor e is 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.