QoS-Aware Shared Component Composition for Distributed Stream Processing Systems

Thomas Repantis, Member, IEEE, Xiaohui Gu, Member, IEEE, and Vana Kalogeraki, Member, IEEE

IEEE Transactions on Parallel and Distributed Systems, 20(7), 968-982
Introduction - Distributed Stream Processing Systems

Distributed Stream Processing Systems

- High-volume, continuous input streams
- Processed result streams

Component

On-line processing functions / continuous query operators implemented on each node:

- Clustering
- Correlation
- Filtering
- Aggregation
- ...
Motivation

• In **DSPSs (Distributed Stream Processing Systems)**, streams **continuously** arrive components, components need to process input streams **in real time** to generate output streams.

**Major challenge:**

Select among different component to compose stream processing applications **on demand**.
Motivation

• focuses on enabling sharing-aware component composition for efficient distributed stream processing.

• Sharing-aware composition allows different applications to utilize:
  • previously generated streams
  • already deployed stream processing components
System model - Synergy Architecture

Fig. 2. Synergy system architecture.
Algorithm - Synergy component composition protocol

**Input:** query $\langle \xi, Q_\xi, \rangle$, node $v_s$

**Output:** application component graph $\lambda$

---

Fig. 5. Query plan example.
Algorithm - Synergy component composition protocol

1. $v_S$ identifies maximum sharable point(s) in $\xi$
2. $v_S$ spawns initial probes

![Diagram of component composition protocol]

Fig. 3. Probing example.

Probing path: 1. S -> C1 -> C3 -> D
   2. S -> C1 -> C4 -> D
   3. S -> C2 -> C3 -> D
   4. S -> C2 -> C4 -> D
Algorithm - Synergy component composition protocol

3 for each $v_i$ in path
4 checks available resources
5 AND checks QoS so far in $Q_\xi$
6 AND checks projected QoS impact

Fig. 3. Probing example.
Algorithm - Synergy component composition protocol

7    if probed composition qualifies
8 sends acknowledgement message to upstream node
9 performs transient resource reservation at \( v_i \)
10 discovers next-hop candidate components from \( \xi \)
11 deploys next-hop candidate components if needed
12 spawns probes for selected components
13 else drops received probe
Algorithm - Synergy component composition protocol

14 $v_s$ selects most load-balanced component composition $\lambda$
15 $v_s$ establishes stream processing session
Experimental Evaluation - Setup

- Implemented as a multithreaded system including about 20,000 lines of Java code
- Running on each of 88 physical nodes of PlanetLab.
- Based on the SpiderNet service composition framework.
- One hundred components were deployed uniformly across the nodes, with a replication degree of 5.
- Application requests asked for two to four components chosen randomly and for the corresponding streams between the components.
- Generate approximately nine requests per second throughout the system, using a Zipf distribution with $\alpha = 1.6$
Experimental Evaluation - Setup

- Compared Synergy against two different composition algorithms:
  1. A Random algorithm that randomly selected one of the candidates for each application component
  2. A Composition algorithm performs QoS-aware composition but does not consider result stream reuse or component reuse
Experimental Evaluation

Average application end-to-end delay

Successful application requests
Experimental Evaluation

Protocol overhead

Breakdown of average setup time

<table>
<thead>
<tr>
<th>Setup Time (ms)</th>
<th>Random</th>
<th>Composition</th>
<th>Synergy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery</td>
<td>240</td>
<td>188</td>
<td>243</td>
</tr>
<tr>
<td>Probing</td>
<td>4509</td>
<td>4810</td>
<td>3141</td>
</tr>
<tr>
<td>Total</td>
<td>4749</td>
<td>4998</td>
<td>3384</td>
</tr>
</tbody>
</table>
Conclusions

• Synergy:
  • built on top of a totally decentralized overlay architecture
  • reuse existing streams and components
  • ensure that the QoS requirements of the currently running applications

• Prototype implementation of Synergy over PlanetLab shows that:
  • sharing-aware component composition can enhance QoS provisioning for distributed stream processing applications.