Hot-Spot Prediction and Alleviation in Distributed Stream Processing Applications

Thomas Repantis       Vana Kalogeraki

Department of Computer Science & Engineering
University of California, Riverside
{trep,vana}@cs.ucr.edu
http://www.cs.ucr.edu/~{trep,vana}/
On-Line Data Stream Processing

Network traffic monitoring for intrusion detection

Analysis of readings coming from sensors or mobile robots

Click stream analysis for purchase recommendations or advertisements

Customization of multimedia or news feeds

Thomas Repantis
Distributed Stream Processing System

High volume data streams (sensor data, financial data, media data)

Extracted result streams

Real-time online processing functions/Continuous query operators

Filter  Correlation  Clustering  Aggregation

Thomas Repantis
Streams are processed online by components distributed across hosts.

Data arrive in large volumes and high rates, while workload spikes are not known in advance.

Stream processing applications have QoS requirements, e.g., e2e delay.

Thomas Repantis
QoS for Distributed Stream Processing Applications

Our goal: How to avoid QoS violations in distributed stream processing applications at run-time
- Predict QoS violations
- Alleviate hot-spots

Challenges
- Burstiness
  - Data rates may increase suddenly due to external events
- Highly dynamic environment
  - Application requests arrive on-demand
- Scale that dictates decentralization
  - Global state changing too fast for a single node to follow
Application-Oriented Load Management

- Application hot-spots: QoS violations
  - Sensitive hot-spot detection
    - Triggered even on moderately loaded nodes, if stringent QoS
  - Fine-grained hot-spot alleviation
    - Only suffering applications migrate
  - Proactively prevent QoS degradation
    - Focus on application performance rather than system resources
Roadmap

- Motivation and Background
- Synergy Architecture
- Design and Algorithms
  - Hot-Spot Prediction
  - Hot-Spot Alleviation
- Experimental Evaluation
- Related Work
- Conclusion
Synergy Middleware

- A middleware managing the mappings:
  - From application layer to stream processing overlay layer
  - From stream processing overlay layer to physical resource layer
Synergy Node Architecture

- Application Composition and QoS Projection instantiate applications
- Replica Placement places components
- Load Balancing and Load Prediction detect hot-spots
- Migration Engine alleviates hot-spots
- Monitor measures processor and bandwidth
- Discovery locates streams and components
- Routing transfers streaming data
System Model

Application parameters:
- **QoS** $q_t$: User requested end-to-end execution time
- **End-to-end execution time**: $\sum_{i=1}^{n} t_{e(i)} + \sum_{i=1}^{n} t_{c(i)}$
- **Slack time** $t_s$: Time left before violating QoS $q_t$

Component parameters:
- **Execution time** $t_e$: Time to process a stream tuple
- **Communication time** $t_c$: Time to transfer a stream tuple
- **Component rate** $r_k$: Input stream rate
Predicting QoS Violations

- Positive slack time $t_s$ for every application component means no QoS violation
- Calculate $t_s$ on every component $i$ based on execution time $t_e$ and communication time $t_c$

$$t_s(i) = q_t - \left( \sum_{j=1 \ldots i-1} t_{c(j \rightarrow j+1)} + \sum_{j=1 \ldots i-1} t_{e(j)} + \sum_{i \ldots v-1} t_{c(j \rightarrow j+1)} + \sum_{j \in i \ldots v} \hat{t}_{e(j)} \right) > 0$$

Thomas Repantis
Execution Time Prediction

- Rate fluctuations and bursts common in stream processing applications
- High arrival rates make prediction per tuple more fine-grained than node load changes
- We use linear regression to bind execution time $t_e$ and total rate $r_t$

\[
\hat{t}_e = a + b \cdot \hat{r}_t
\]

\[
a = \bar{t}_e - b \cdot \bar{r}_t
\]

\[
b = \frac{\sum_{j=1}^{k} (r_{t(j)} - \bar{r}_t) \cdot (t_{e(j)} - \bar{t}_e)}{\sum_{j=1}^{k} (r_{t(j)} - \bar{r}_t)^2}
\]
Rate Prediction

- **Auto-correlation (capture self-similarity)**
  
  \[
  \hat{r}_k = \frac{\arg_m \max R(k)}{\arg_m \max R(k-1)} \cdot r_{k-1} = \frac{r_k(m)}{r_{k-1}(m)} \cdot r_{k-1}
  \]

- **Cross-correlation (Pearson Product Moment) (capture components in same application)**
  
  \[
  R_i = \frac{\sum_{j=1}^{k-1} (r_j(i) - \bar{r}_i)(r_j - \bar{r})}{\sqrt{\sum_{j=1}^{k-1} (r_j(i) - \bar{r}_i)^2 \sum_{j=1}^{k-1} (r_j - \bar{r})^2}}
  \]

*Thomas Repantis*
Metadata Layer Over a DHT

- Decouples component placement from their discovery
- Component names are hashed in a DHT
- DHT maps the hashed names to nodes currently offering the specified component
Decentralized Load Monitoring

- DHT maps component names to the loads of peers hosting them
- Peers autonomously detect hot-spots

- Load updates pushed when intervals change
- Overlapping intervals absorb frequent changes

Thomas Repantis
Alleviating Hot-Spots via Migration

Minimum number of migrations to reduce sum of rates in node so that all projected execution times are within QoS requirements.
Experimental Evaluation

- Monitor source-destination pairs in top 5% of total traffic over last 20 minutes [Stream Query Repository]

- TCP traffic trace, LBL, 2 hours, 1.8 million packets [Internet Traffic Archive]

- Synergy multi-threaded prototype on 34 PlanetLab nodes

*Thomas Repantis*
Synergy GUI
Prediction Accuracy

- Average rate prediction error 3.7016%
- Average total prediction overhead 0.5984ms

Thomas Repantis
Overhead and Performance

- Average one migration every three applications
- Average migration time 1144ms

Thomas Repantis
QoS Improvement

As load increases, benefits of hot-spot elimination become evident.

Thomas Repantis
Related Work

- **Component placement**
  - SBON [ICDE'06], SAND [VLDB'04]

- **Component composition**
  - SpiderNet [ICDCS'05], sFlow [ICDCS'04]

- **Load balancing**
  - Borealis [ICDE'05], Flux [ICDE'03], PeerCQ [TC'05]

- **Load shedding**
  - Borealis [VLDB'03], RTStream [RTSS'06]

- **Application adaptation**
  - IFLOW [ICDCS'05], River [TOCS'03]

- **Resource load prediction**
  - RPS [TPDS'06], Regression [ICAC'07], Mmodel [CCGRID'04]

*Thomas Repantis*
Conclusion

▷ Synergy: QoS-Enabled Distributed Stream Processing Middleware
  ▷ Predict QoS violations at run-time
  ▷ Alleviate hot-spots using migration

▷ Future work
  ▷ Multiple shared resources in prediction framework
  ▷ State transfer when migrating complex components

Thomas Repantis
Hot-Spot Prediction and Alleviation in Distributed Stream Processing Applications

Thomas Repantis    Vana Kalogeraki
{trep,vana}@cs.ucr.edu
http://www.cs.ucr.edu/~{trep,vana}/
synergy
http://synergy.cs.ucr.edu/