Designing High Fan-in Systems

Presented by: Yi Zhang
Partial content adapted from presentation by S. Jeffery at CIDR conference, January 2005, Asilomar

Outline
- Background and Motivation
- HiFi Query Processing
- Initial Architecture and Prototype
- Discussion

Receptors Everywhere

- Receptors in life:
  - Wireless sensor networks, RFID (Radio-frequency Identification), cameras,…
  - Widely distributed large scale system
  - Support real-time monitoring, detecting, planning, and decision making

The HiFi Bowtie

Design Challenge: Time

- Filtering, Cleaning, Alerts
- Monitoring, Time-series
- Data mining
- Archiving

seconds | Time Scale | years
On-the-fly processing | Stream/Disk Processing | Disk-based processing

Design Challenge: Space

- Filtering, Cleaning, Alerts
- Monitoring, Time-series
- Data mining
- Archiving

local | Space Scale | global
Shelf | Regional Centers | Headquarter
Design Challenge: Resources

Filtering, Cleaning, Alerts
Monitoring, Time-series
Data mining
Archiving

Computing Resource Scale

tiny

huge

Case Study: Supply Chain Mgmt

Headquarters
Regional Centers
Warehouses
Warehouse Doors
Receptors

Why HiFi?

• Current approaches: hand-coded, script-based
  – Expensive, one-shot, redundant work
• Piecemeal systems
  – Each type of receptor handled separately
• No uniform framework incorporating different data models, protocols, etc.
  – Each level presents its own API

Outline

• Background and Motivation
• HiFi Query Processing
• Initial Architecture and Prototype
• Discussion

HiFi

• A data management infrastructure for high fan-in environments
• Uniform declarative framework
  – Stream-oriented
  – SQL-like query language
  – Data flows from the edges inwards: cascading streams

CSAVA: RFID Data Processing

• Erroneous, lost, duplicate data

Clean
Smooth
Arbitrate
Validate
Analyze
"I know that the warehouse in Springfield should have 1,000 widgets."

Outline

- Background and Motivation
- HiFi Query Processing
- Initial Architecture and Prototype
- Discussion
Processing Queries

- Dispatch query $Q$ to its root
- Transform $Q$ into compatible repr. with views
- Rewrite $Q$ for children views: $Q_1, \ldots, Q_i$
- Create new stream definition for each $Q_i$
- Decide where and when to run $Q_i$
- Disseminate queries
- Receive result data and stream into DSP
- Return results to appropriate parent(s)

Initial Prototype

- TelegraphCQ
  - Data stream processor
  - Continuous, adaptive query processing with aggressive sharing
- TinyDB
  - Declarative query processing for wireless sensor networks
  - In-network aggregation
Discussion

- Loss of flexibility
  - Data representation: streams, queries, views
  - Fixed tasks in each level: does everybody need cleaning, smoothing, etc.?
  - Targeted at specific applications (RFID)
- Do we really need this complex, unified framework?
  - Why force receptors and PCs present the same set of API?
  - Too much burden for low level nodes (running 12 kinds of services provided by HiFi Glue!)

Outline

- Background and Motivation
- HiFi Query Processing
- Initial Architecture and Prototype
- Discussion

Q&A

Thank you!