Exploiting Constraints to Build a Flexible and Extensible Data Stream Processing Middleware

Workshop on Scalable Stream Processing Systems (SSPS)
IPDPS 2010, Atlanta/Georgia, USA
19.04.2010

Nazario Cipriani, Carlos Lübbe, Alexander Moosbrugger
Context-aware applications need context data to be processed
- Context-aware systems utterly heterogeneous
- Nearby sensors produce context data streams that needs to be processed online
- Not convenient to first store and then process offline due to high data volume
  \[\Rightarrow\] Use *Data Stream Processing Systems (DSPS)*!

Data streams are handled by Data Stream Processing Systems
- A.k.a. Data Stream Management Systems (*DSMS*)
- Data streams are processed according to some processing definition
- Push-based processing paradigm instead of pull-based

Today's DSPS offer a broad range of sophisticated and efficient processing schemes for online data stream processing
- Well suited for general purpose data stream processing
Agenda

- Non-Trivial Application Scenario of a Distributed Visualization Pipeline

- Adaptation Problem for non-trivial Applications
  - Gap Between Application’s and System’s Interests

- Constraints Classification

- Enhancing the NexusDS Platform by Constraints
  - NexusDS Platform Overview
  - Network Groups and Operator Model
  - Constraint-Based Data Flow Graphs
  - Enhanced Processing Model Supporting Constraints

- Conclusion and Future Work
Sample Scenario – Distributed Visualization Pipeline

**Sources**

- **Data Stream:** Bus, Taxi, and User Locations
- **Static Data:** Buildings & Streets, Points of Interest

**Operators**

- **Merge & Filtering:** Selection of Nearby Objects
- **Mapping:** Rectangular Map with 3D Buildings
- **Rendering:** Projection Rasterize

**Sinks**

- **Client:** Image Stream (e.g., Smartphones, PDAs, etc.)

**Visualization-Pipeline**

- User Location Updates
- Parameter Updates
Sample Scenario – Distributed Visualization Pipeline

**Sources**

- **Data Stream:** Bus, Taxi, and User Locations
- **Static Data:** Buildings & Streets, Points of Interest

**Operators**

- **Merge & Filtering:** Selection of Nearby Objects
- **Mapping:** Rectangular Map with 3D Buildings
- **Rendering:** Projection Rasterize

**Sinks**

- **Client:** Image Stream (e.g., Smartphones, PDAs, etc.)

**Built-In Operator** can be used here ➔ **Exploit already existing operators**!

**Custom Operator** to define the **Mapping Process** for Data to Rendering Primitives!

Dedicated hardware can improve performance ➔ **Exploit specialized hardware is reasonable**!
Sample Scenario – Distributed Visualization Pipeline

**Sources**
- Data Stream: Bus, Taxi, and User Locations
- Static Data: Buildings & Streets, Points of Interest

**Operators**
- Merge & Filtering: Selection of Nearby Objects
- Mapping: Rectangular Map with 3D Buildings
- Rendering: Projection Rasterize

**Sinks**
- Client: Image Stream (e.g., Smartphones, PDAs, etc.)

Parameter Updates

User Location Updates

Visualization-Pipeline

Mobile devices still have limited capabilities (energy) ➔ **Outsource processes if possible!**
Requirements for DSPS to Support Non-Trivial Applications?

- **Extensible operator base**
  - Complex domain specific operators, may require dedicated hardware
  - Support for structured and unstructured data: “Tuples” vs. images, videos

- **Heterogeneous system topology**
  - System with wide range of devices in mind – powerful computing servers vs. mobile devices
  - Exploit available hardware for efficient execution

- **Operator has deployment and runtime restrictions**
  - Influence deployment and runtime of operators in processing pipelines
Adaptation Gap between Applications and DSPS

- Many different (distributed) context data streams ➞ Use distributed DSPS

- Applications or domains may ask for specific functionality
  - But still depending on general stream processing principles
  ➞ Reuse existing components, “do not reinvent the wheel”!
Adaptation Gap between Applications and DSPS

- Many different (distributed) context data streams ➔ Use distributed DSPS

- Applications or domains may ask for specific functionality
  - But still depending on general stream processing principles
    ➔ Reuse existing components, “do not reinvent the wheel”!

![Diagram showing the adaptation problem between Data Stream Processing System (DSPS) and Domain-specific Data Stream Application. Adapt DSPS to Application (1) and Adapt Application to DSPS (2).]
Adaptation Gap between Applications and DSPS

- Many different (distributed) context data streams ➞ Use distributed DSPS

- Applications or domains may ask for specific functionality
  - But still depending on general stream processing principles
  ➞ Reuse existing components, “do not reinvent the wheel”!

Diagram:
- Data Stream Processing System (DSPS)
- Domain-specific Data Stream Processing System
- Domain-specific Data Stream Application
- Adapt DSPS to Application
- Adapt DSPS to Application
- Adapt Application to DSPS
- Adapt Application to DSPS
- Adaptation Problem
**Constraints Classification**

- **System-Relevant Constraints**
- **Domain-Relevant Constraints**
- **Application-Relevant Constraints**
- **User-Relevant Constraints**

**Deployment Constraints**

**Runtime Constraints**

**Freely Modifiable**

**Bounded Modifiable**
NexusDS Platform Overview

Nexus Applications and Extensions
- Application Operators
- Application Services

Nexus Domain Extensions
- Domain Operators
- Domain Services

Nexus Core
- Core Operators
- Core Query Service (CQS)
- Operator Repository Service (ORS)
- Operator Execution Service (OES)

Communication and Monitoring
- Monitoring Service (MS)
- Service Publisher Service (SPS)
NexusDS Platform Overview

- User-Relevant Constraints
- Application-Relevant Constraints
- Domain-Relevant Constraints
- System-Relevant Constraints

- Nexus Applications and Extensions
  - Application Operators
  - Application Services
- Nexus Domain Extensions
  - Domain Operators
  - Domain Services
- Nexus Core
  - Core Operators
  - Core Query Service (CQS)
  - Operator Repository Service (ORS)
  - Operator Execution Service (OES)
- Communication and Monitoring
  - Monitoring Service (MS)
  - Service Publisher Service (SPS)
Classification of Runtime Environments

- Organization scheme of available environments to reduce deployment complexity
- Access corresponding groups to get available execution environments

NEXUS

- Nexus Group
  - Core Group
  - Peer Properties Group
  - Visualization Group
  - Arbitrary Domain Group
  - Arbitrary Domain Subgroup

Hardware
- x86 CPU Architecture
- ATI RV730 GPU
- ...

Software
- Windows
- JVM 1.5
- ...

User-Relevant
Application-Relevant
Domain-Relevant
System-Relevant
Operator Packages – Overview

- **Meta Data**
  - Defines operator’s properties
  - Descriptor
  - Requirements → Deployment
  - Presets → Runtime

- **Program Code**
  - Actual operator implementation

- **Dependencies**
  - Third party libraries
Constraint-Based Data Flow Graphs

**Deployment**
- Name = Mapping
- Author = Me
- NodeID = 007…009
- OperatorID = 123
- Hardware = GPU

**Data Stream:**
- Bus, Taxi, and User Locations

**Merge & Filtering:**
- Selection of Nearby Objects

**Static Data:**
- Buildings & Streets
- Points of Interest

**Mapping:**
- Rectangular Map with 3D Buildings

**Client:**
- Image Stream
  - e.g., Smartphones, PDAs, etc.

**Deployment**
- Client: Image Stream

**Runtime**
- Buffer = 10
- Resolution = 800x600
- FPS = 20
- Buffer = 100
- Model_Detail = Medium

**Rendering:**
- Projection Rasterize

**Deployment**
- NodeID = 007…009
- OperatorID = 123
- Hardware = GPU

**Deployment**
- Client: Image Stream
  - e.g., Smartphones, PDAs, etc.

**Runtime**
- NodeID = 007…009
- OperatorID = 123
- Hardware = GPU

**Deployment**
- Client: Image Stream
  - e.g., Smartphones, PDAs, etc.

**Runtime**
- Buffer = 10
- Resolution = 800x600
- FPS = 20
- Buffer = 100
- Model_Detail = Medium

**Deployment**
- NodeID = 007…009
- OperatorID = 123
- Hardware = GPU

**Deployment**
- Client: Image Stream
  - e.g., Smartphones, PDAs, etc.

**Runtime**
- Buffer = 10
- Resolution = 800x600
- FPS = 20
- Buffer = 100
- Model_Detail = Medium

**Deployment**
- NodeID = 007…009
- OperatorID = 123
- Hardware = GPU

**Deployment**
- Client: Image Stream
  - e.g., Smartphones, PDAs, etc.

**Runtime**
- Buffer = 10
- Resolution = 800x600
- FPS = 20
- Buffer = 100
- Model_Detail = Medium
Create the domain service and operators ➔ Constraints!
Deployment Model – Visualization Pipeline Sample Scenario

User wants a certain resolution, 800x600 → Constraint!

Application wants certain frame rate, 20 FPS → Constraint!

- Nexus Application
- Nexus Domain Extension
- Nexus Core
- Communication and Monitoring
- Physical Layer
  - UMPC Client (Intel A110 @ 800MHz, 1GB RAM)
  - GPU-enabled PC (AMD X2 @ 1,6GHz, 2GB RAM, Nvidia-GF GO 7200)
  - PC (AMD X2 @ 1,6GHz, 2GB RAM)
- Operator Repository Service (ORS)
- Operator-Repository-Service (ORS)

- Visual Client (VC)
- Rendering Step (Operator)
- Mapping Step (Operator)
- Filtering & Merge Step (Operator)
- Mobile Objects & Buildings (Sources)
- Client Application (Sink)
- Core Query Service (CQS)
- Operator Execution Service (OES)

User-Relevant
Application-Relevant
Domain-Relevant
System-Relevant
Deployment Model – Visualization Pipeline Sample Scenario

Only well known (secure) Nodes should be used! ➔ Constraint!

User-Relevant
Application-Relevant
Domain-Relevant
System-Relevant

Nexus Application
Nexus Domain Extension
Nexus Core
Communication and Monitoring
Physical Layer

Operational Repository Service (ORS)

Visualization Pipeline
Service (VPS)

Core Query Service (CQS)

Operator Execution Service (OES)

Operator Repository Service (ORS)

Visual Client (VC)
Client Application (Sink)

Rendering Step (Operator)

Mapping Step (Operator)

Filtering & Merge Step (Operator)

Mobile Objects & Buildings (Sources)

Buildings, Streets, POIs

Bus Locations

Only well known (secure) Nodes should be used!
Deployment Model – Visualization Pipeline Sample Scenario

At least one node (Render) with **GPU** needed! → Constraint!

All constraints must be checked against available resources!

User-Relevant

Application-Relevant

Domain-Relevant

System-Relevant

Nexus Application

Nexus Domain Extension

Nexus Core

Communication and Monitoring

Physical Layer

**Physical Layer**

- **UMPC Client** (Intel A110 @ 800MHz, 1GB RAM)
  - **Rendering Step (Operator)**
- **GPU-enabled PC** (AMD X2 @ 1,6GHz, 2GB RAM, Nvidia-GF GO 7200)
  - **Mapping Step (Operator)**
- **PC** (AMD X2 @ 1,6GHz, 2GB RAM)
  - **Filtering & Merge Step (Operator)**
- **GPU-enabled PC** (AMD X2 @ 1,6GHz, 2GB RAM, Nvidia-GF GO 7200)
  - **Monitor Service (MS)**
- **Visual Client** (VC)
- **Visual Client Application (Sink)**
- **Operator Repository Service (ORS)**
- **Operator Execution Service (OES)**

**Domain-Relevant**

**User-Related Application**

**Application-Related Domain**

**System-Related Relevant**

**Extension Developer**

**Extension**

**Constraint!**

**Constraint!**

**Constraint!**
Deployment Model – Visualization Pipeline Sample Scenario

- **Nexus Application**
- **Nexus Domain Extension**
- **Nexus Core**
- **Communication and Monitoring**
- **Physical Layer**

**Secure!**

- **Visualization Pipeline Service (VPS)**
- **Core Query Service (CQS)**
- **Operator Execution Service (OES)**
- **Monitoring Service (MS)**

**Physical Layer**
- **UMPC Client (Intel A110 @ 800MHz, 1GB RAM)**
- **GPU-enabled PC (AMD X2 @ 1,6GHz, 2GB RAM, Nvidia-GF GO 7200)**
- **PC (AMD X2 @ 1,6GHz, 2GB RAM)**
- **Buildings, Streets, POIs**
- **Bus Locations**

**Visual Client (VC)**

**Visual Client Application (Sink)**

**Rendering Step (Operator)**

**Mapping Step (Operator)**

**Filtering & Merge Step (Operator)**

**GPU!**

**User-Relevant**

**Application-Relevant**

**Domain-Relevant**

**System-Relevant**
Deployment Model – Visualization Pipeline Sample Scenario

Nexus Application

Nexus Domain Extension

Nexus Core

Communication and Monitoring

Physical Layer

Visualization Pipeline Service (VPS)

Core Query Service (CQS)

Operator Execution Service (OES)

Monitoring Service (MS)

User-Relevant

Application-Relevant

Domain-Relevant

System-Relevant

4.2

Visualization Pipeline Service (VPS)

Core Query Service (CQS)

Operator Execution Service (OES)

Monitoring Service (MS)

Physical Layer

UMPC Client (Intel A110 @ 800MHz, 1GB RAM)

GPU-enabled PC (AMD X2 @ 1,6GHz, 2GB RAM, Nvidia-GF GO 7200)

PC (AMD X2 @ 1,6GHz, 2GB RAM)

Buildings, Streets, POIs

Bus Locations

Client Application (Sink)

Rendering Step (Operator)

Mapping Step (Operator)

Filtering & Merge Step (Operator)

Visual Client (VC)

Visual Client (VC)

Visualization Pipeline Service (VPS)

Core Query Service (CQS)

Operator Execution Service (OES)

Monitoring Service (MS)

Mobile Objects & Buildings (Sources)
Conclusion and Future Work

- Adaptation problem for non-trivial applications

- Identified requirements to satisfy needs of specific applications
  - Based on non-trivial example of distributed visualization pipeline
  - Heterogeneous system topologies, highly domain-specific operators, operators connect their execution to explicit runtime and deployment restrictions
    - Constraints!

- Deployment is done according to pre-defined requirements on different levels (constraints)

- Future things to do
  - Create suitable constraint-definition language to express complex constraint-links
  - Optimize deployment and execution process in terms of costs

Finish... And Flowers for You!