Adaptive Application Performance Management for Big Data Stream Processing

Holger Eichelberger, Cui Qin, Klaus Schmid, Claudia Niederée

{eichelberger, schmid,qin}@sse.uni-hildesheim.de
niederee@L3S.de
Adaptive Performance Management for Big Data Stream Processing

Contents

• Motivation
• Performance Management Problem
• Our Approach
• Summary / Outlook
Adaptive Performance Management for
Big Data Stream Processing

Motivation

- Big Data
  - Processing of large and complex data sets
  - Too difficult for traditional data processing applications
  - 3V: Volume, Velocity, Volatility
- Risk identification in financial markets (FP7 QualiMaster)
  - Interconnected markets
  - Regular risk analysis requested by EU / US law
  - Bursty data streams
    - Financial data
    - Social web
Adaptive Performance Management for Big Data Stream Processing

Performance Management Problem

Problem (Performance View)

- Data processing pipeline
  - Soft real-time constraints
- Varying stream characteristics
  - Several orders of magnitude
- Resource pool
  - Constrained
  - Specialized hardware (e.g., FPGA)

- **Goal:** Simplified Development of Performance Management Mechanisms
  - Lightweight for the developer
  - Resource-aware configuration
  - Model-based generation
  - Adaptive Performance Management
Adaptive Performance Management for
Big Data Stream Processing

Tradeoffs

- Soft real-time constraints
  - Processing latency ↔ Utility of results
  - Number of events may vary, latency shall not
- Resource constraints
  - Optimal allocation to available resources
  - Heterogeneous resource pool
  - Minimize resource costs
- Result precision
  - Algorithms offer different precision
  - Algorithms differ in performance, resource usage
Main concepts

• Algorithm families
• Adaptive data analysis pipelines

- Pipeline adaptation
  - Select most appropriate algorithm
  - Modify algorithm parameters
  - Resource allocation
  - Change structure of pipeline
Approach

Application Development

- **Configure** the application
  - Topological configuration
  - Complex constraints
- **Validate** the configuration
- **Generate** the implementation
  - Bind Pipeline and Execution Infrastructure
    - Apache Storm
    - Maxeler Data Flow Engines
  - Introduce algorithm switching and monitoring probes
- **Deploy and run**
Adaptive Performance Management (1)

- MAPE-K: Monitoring, Analysis, Planning, Execution – Knowledge

- Monitoring
  - Statistics by Apache Thrift
    - Execution time, processed items, executors
  - SPASS-meter
    - Memory, network, file transfer
  - Generated monitoring probes
  - Hardware: Available FPGAs
  - Derived: Capacity, pipeline measures

Trend: More generated probes!
Adaptive Management (2)

• Analysis
  • Constraint-based deviations from current behavior
  • Watermarking-scheme for resource usage

• Planning
  • Determine changes to the runtime configuration
  • Configuration + adaptive planning
  • Basis: Stitch, S/T/A
  • Actions modify configuration and can generate new code

Future: Profiles and Predictions!
Adaptive Management (3)

• Execution
  • Enact changes due to runtime configuration
  • Coordination: Software vs. Hardware Execution
  • Examples:
    • Change parallelization at runtime
      • Apache Storm: 8 s stop
      • Modified Storm: < 30 ms stop
    • Switch distributed algorithms
      • Naïve: 23 s
      • Improved < 50 ms + queue transfer

Future: State transfer!
Adaptive Performance Management for Big Data Stream Processing

Conclusions

• Simplification of adaptive performance management
  • Lightweight for the developer
  • Configure, validate, generate
  • Adaptive management through MAPE-K
• Adaptive performance management is a challenge
  • Gap-free enactment
• Future
  • More detailed experiments
  • Offline / online algorithm profiles
  • Generic vs. application-specific (generated) probes