Scheduling Decisions in Stream Processing on Heterogeneous Clusters

Marek Rychlý, Petr Škoda, Pavel Smrž

Department of Information Systems
Faculty of Information Technology
Brno University of Technology
(Czech Republic)

The 8th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2014)
2 – 4 July, 2014

The 4th Semantic Web/Cloud Information and Services Discovery and Management (SWISM 2014)

Outline

Processing of Big Data

- Big Data are large and complex data sets
  (too large/complex to manipulate or interrogate with standard methods or tools)
- processing is performed in distributed parallel architectures
  (data distributed across multiple nodes which can process the data in parallel)
- batch and stream processing = data-sets and data-streams
  data-sets ⇒ batch processing, MapReduce paradigm
  (e.g., Apache Hadoop, Java 8 Streams)
  data-streams ⇒ stream processing
  (e.g., Aurora*/Medusa et al., Hadoop Online, Apache Storm)
- response times in batch processing typically greater than 30 sec, response times in real-time stream processing in (sub)seconds
  (data streams are usually continuous, they have to be processed in real-time)

Heterogeneous Clusters

- clusters heterogeneous in resources available for computation
- different types of nodes, various computing performance and capacity (high-performance nodes can complete the processing of identical data faster)
- performance depends on the character of computation and input data (graphic-intensive computations will run faster on nodes with powerful GPUs, particular algorithms can be accelerated by FPGAs, etc.)
- performance characteristics of individual nodes
  1. defined at design-time (infrastructure and topology of a cluster, specification of its individual nodes)
  2. measured at run-time (performance monitoring data and their statistical analysis of processing different types of computations and data by different types of nodes)

Scheduling in Distributed Stream Processing

- scheduling which tasks and when to place on which allocated resources (assigning resources to requesters is the responsibility of resource allocation)

  = mapping instances of the tasks to resources provided by a cluster (e.g., to cluster nodes of various types and performance)
- in batch processing, it can be done prior to the processing of a batch (based on knowledge of resources, data and tasks for processing)
- in stream processing, it has to be done at run-time and often in real-time (based on actual intensity of input data flow, quality of the data, and on workload)

$\Downarrow$

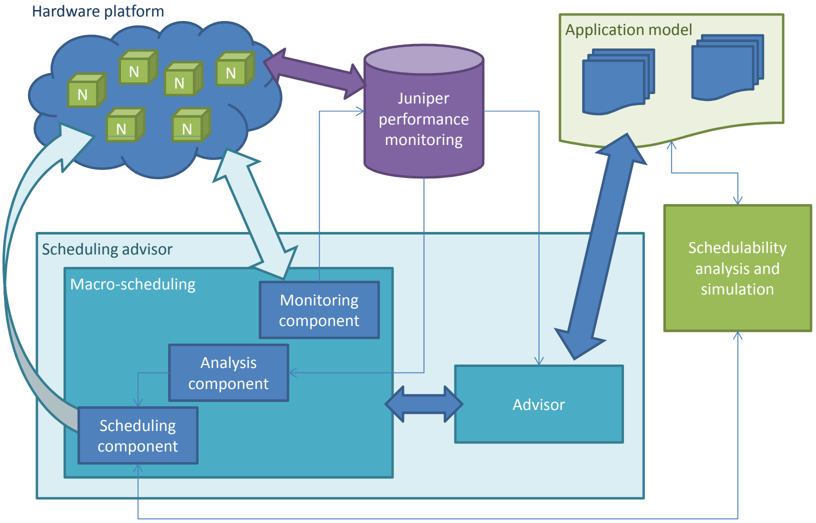
benchmarking of a cluster and profiling of an application at runtime to the best placement of application task instances to cluster nodes

Scheduling Advisor in the JUNIPER Platform

- JUNIPER[1] platform for distributed stream processing
  (based on MPI, Java 8 Streams, and Apache Hadoop infrastructure)
- a novel scheduling advisor which provides scheduling decisions
  (tota scheduler which performs scheduling in the JUNIPER platform)
- the advisor utilizes a programming model of an application
  (the model describes the applications' architecture, incoming and outgoing streams, internal streams, real-time constraints for individual tasks, etc.)
- the advisor performs benchmarking and profiling at runtime
  (benchmarking of a cluster/platform and profiling of an application)

[1] Java platform for hIgh PErformance and Real-time large scale data management, `http://www.juniper-project.org/`

Scheduling Advisor in the JUNIPER Platform



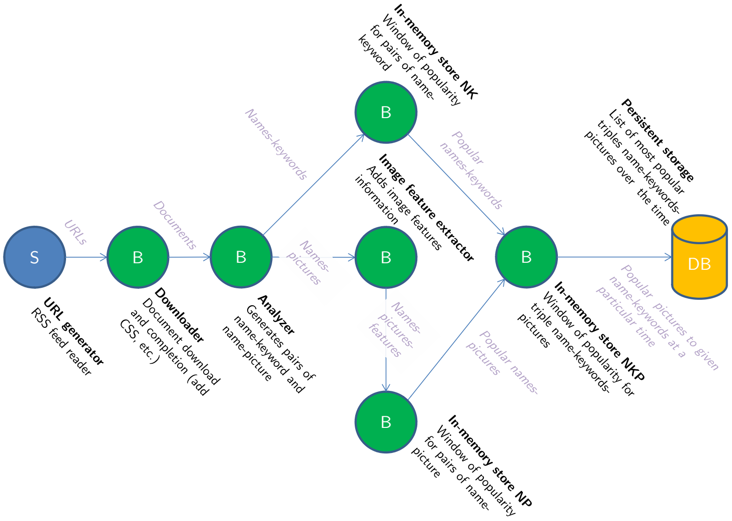
Benchmarking & Profiling by the Scheduling Advisor

1. at first run of the application, the advisor performs a random placement[2], (i.e., random mapping of the tasks instances to the platform nodes)
2. then, it concurrently performs both profiling of the tasks and benchmarking of the platform nodes, (to measure performance of particular task on particular platform nodes)
3. it repeatedly performs and measures different placements with various assignments of the tasks on particular platform nodes, (to get new profiling/benchmarking data on yet unobserved placements)
4. it starts to optimize deployment as soon as
   - no new data can obtainable by further profiling/benchmarking, (all possible task-to-node mappings have been profiled/benchmarked)
   - there is an instantaneous need for the most optimal deployment according to the current profiling and benchmarking data, (e.g., the application goes into production)

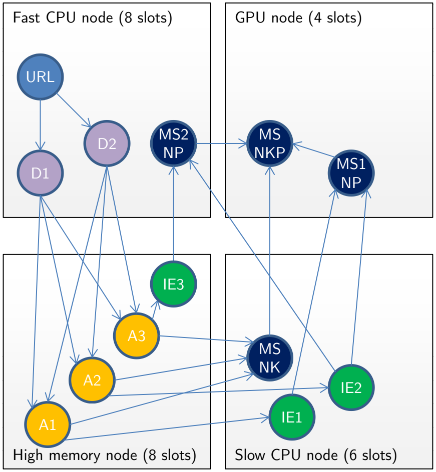
[2] the placement is not entirely random, the design-time knowledge of the application's model is utilized, e.g., to meet real-time constraints defined in the model

A Sample Application for Stream Processing

Spouts and Bolts components in the application's topology for Apache Storm


S
URL generator
RSS feed reader
URLs
B
Downloader
Document download and completion (add CSS, etc.)
Documents
B
Analyzer
Generates pairs of name-keyword and name-picture
Names-keywords
Names-pictures
B
In-memory store NK
Window of popularity for pairs of name-keyword
B
Image feature extractor
Adds image features information
Names-pictures-features
B
In-memory store NP
Window of popularity for pairs of name-picture
Popular names-keywords
Popular names-pictures
B
In-memory store NKP
Window of popularity for triple name-keywords-pictures
Popular pictures to given name-keywords at a particular time
DB
Persistent storage
List of most popular triples name-keywords-pictures over the time

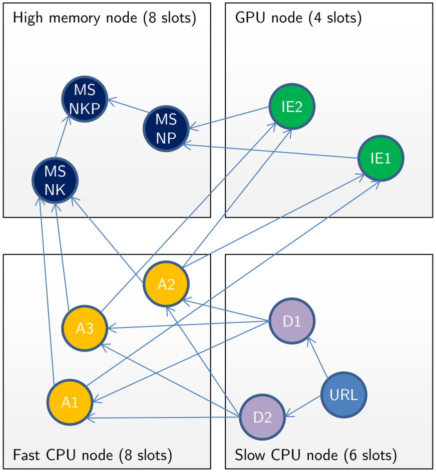
Placement of Tasks by the Standard Scheduler



URL—URL generator; Dx—Downloader; Ax—Analyzer; IEx—Image feature extractor; MSx—In-memory store

D/URL	the fast CPU node runs the undemanding Downloaders and the URL generator task
A. . .	Analyzers tasks, which require the CPU performance, were placed to the node with lots of memory
MS. . .	while the memory greedy In-memory stores tasks were scheduled to the nodes with powerful GPU and slow CPU
IE. . .	Image extractor tasks were placed to the slow CPU node and the high memory node

Placement of Tasks by the Scheduling Advisor



URL—URL generator; Dx—Downloader; Ax—Analyzer; IEx—Image feature extractor; MSx—In-memory store

MS. . .	In-memory store tasks were deployed on the node with high amount of memory
IE. . .	Image feature extractor tasks were deployed on the node with two GPUs (it was possible to reduce parallelism)
A. . .	Analyzers utilize Fast CPU node
D/URL	undemanding Downloaders and URL generator tasks were placed on the Slow CPU node

Performance Comparison

for the placements with and without help of the scheduling advisor

Component	**W tuples**	**S tuples**	**B tuples**	**S-W gain**	**B-S gain**	**B-W gain**
AnalyzerBolt	135096	163993	164714	121,39 %	100,44 %	121,92 %
DownloaderBolt	1396	1499	1494	107,38 %	99,67 %	107,02 %
ExtractFeaturesBolt	39745	41867	47991	105,34 %	114,63 %	120,75 %
FeedReaderBolt	1580	1576	1576	99,75 %	100, %	99,75 %
FeedUrlSpout	45711	46334	45654	101,36 %	98,53 %	99,88 %
IndexBolt	39744	41866	47989	105,34 %	114,63 %	120,75 %
Total	**263272**	**297135**	**309418**	**112,86 %**	**104,13 %**	**117,53 %**

W - worst scheduler, S - standard scheduler, B - performance and benchmark based scheduler
S-W gain - gain of standard scheduler over worst scheduler

Summary and Future Work

- Stream processing in heterogeneous clusters requires run-time analysis.
  (benchmarking of a cluster/platform and profiling of an application)
- The run-time analysis by continuous monitoring and re-placement.
  (the particular implementation depend on a platform)
- The scheduling advisor for optimal scheduling decisions in JUNIPER.
  (helps the platform scheduler to optimally utilize heterogeneity of a cluster)

Future work

- more experiments
  (especially in virtualized clusters with high volatile resources)
- better integration with other tools in the JUNIPER project
  (performance analysis provides also hints concerning possible improvements of applications, especially of their architecture/stream processing topology)

Thank you for your attention!

Marek Rychlý

<rychly@fit.vutbr.cz>