Adaptive Resource Management for Distributed Dataflow Systems

Contact: Thomas Renner, Lauritz Thamsen and Odej Kao, Complex and Distributed IT Systems, Technische Universität Berlin, www.cit.tu-berlin.de

Motivation
- Distributed analytic jobs stress different resources
  - Some are CPU intensive, other are I/O intensive
  - Many jobs benefit from local data access
  - Want to select resources for specific performance requirements
- Many jobs are executed repeatedly
  - E.g., daily executed batch jobs and iterative programs
  - Recurring jobs make up 40% of the jobs in clusters

Architecture
- Freamon: job-level cluster monitoring and repository of historical workload data
- Resource utilization, job runtimes, data placements, data access, etc.
- Adaptive resource management tools
  - ① Colocation Assistant: Container and data placement for improved data locality and local data exchange
  - ② Allocation Assistant: Automatic resource allocation for users’ performance targets (e.g. time, utilization)
  - ③ Scaling Assistant: Dynamic adjustment of resource allocations to meet user demands even with unexpected runtime behavior (e.g. data skew, nodes failures)

- Freamon and our resource management tools are integrated with Hadoop, Flink and Spark

Goal
- Adapt resource management to workload characteristics
  - Collect job profile information
  - Gather user performance demands
  - Provide scheduling hints for recurring jobs to
    - improve resource utilization,
    - meet performance targets,
    - and cut operating costs

Results
① Data and Container Colocation
Sets of files that are processed jointly are marked as related and automatically colocated on the same set or subset of nodes. Execution containers are also scheduled on this set of nodes. As a result, this improves data locality and reduces the execution time, especially for data-intensive workloads.

② Runtime-Based Resource Allocation
Users can explicitly express runtime targets instead of having to guess the required resources (e.g. number of containers, cores, memory, etc.). For this, we model the runtime behavior of jobs using either parametric or nonparametric regression depending on the available historical data.

③ Dynamic Scaling of Iterative jobs
We dynamically scale iterative dataflow jobs (e.g. many machine learning or graph algorithms) using the synchronization barriers in-between iterations. In particular, we adapt resource allocations towards utilization targets based on collected system statistics.

Motivation
- Distributed analytic jobs stress different resources
- Some are CPU intensive, other are I/O intensive
- Many jobs benefit from local data access
- Want to select resources for specific performance requirements
- Many jobs are executed repeatedly
- E.g., daily executed batch jobs and iterative programs
- Recurring jobs make up 40% of the jobs in clusters

Architecture
- Freamon: job-level cluster monitoring and repository of historical workload data
- Resource utilization, job runtimes, data placements, data access, etc.
- Adaptive resource management tools
  - ① Colocation Assistant: Container and data placement for improved data locality and local data exchange
  - ② Allocation Assistant: Automatic resource allocation for users’ performance targets (e.g. time, utilization)
  - ③ Scaling Assistant: Dynamic adjustment of resource allocations to meet user demands even with unexpected runtime behavior (e.g. data skew, nodes failures)

- Freamon and our resource management tools are integrated with Hadoop, Flink and Spark

Goal
- Adapt resource management to workload characteristics
  - Collect job profile information
  - Gather user performance demands
  - Provide scheduling hints for recurring jobs to
    - improve resource utilization,
    - meet performance targets,
    - and cut operating costs

Results
① Data and Container Colocation
Sets of files that are processed jointly are marked as related and automatically colocated on the same set or subset of nodes. Execution containers are also scheduled on this set of nodes. As a result, this improves data locality and reduces the execution time, especially for data-intensive workloads.

② Runtime-Based Resource Allocation
Users can explicitly express runtime targets instead of having to guess the required resources (e.g. number of containers, cores, memory, etc.). For this, we model the runtime behavior of jobs using either parametric or nonparametric regression depending on the available historical data.

③ Dynamic Scaling of Iterative jobs
We dynamically scale iterative dataflow jobs (e.g. many machine learning or graph algorithms) using the synchronization barriers in-between iterations. In particular, we adapt resource allocations towards utilization targets based on collected system statistics.

Contact: Thomas Renner, Lauritz Thamsen and Odej Kao, Complex and Distributed IT Systems, Technische Universität Berlin, www.cit.tu-berlin.de

Staying Tuned!