System Support for Cluster-based Data Aggregation

Kai Shen
Dept. of Computer Science, University of Rochester

Data-intensive Online Applications
- Satellite image composition
  - based on sensor data from the satellite
  - projection of sensor values onto a two-dimensional grid and composition of all values mapped to each grid point to generate the associated pixel
- Image convolution
  - enhance spatial features or subdue noises in images: edge detection, gradient detection, smoothing, blurring, and sharpening
- Keyword searching or pattern matching
  - find objects that match input keywords or pattern

- datasets for these applications are often partitioned
- a successful service invocation requires the aggregation of data from many partitions

Data Aggregation
- two ways in general:
  - aggregate the data before processing
  - preprocess the local data and then aggregate the results
- which is better? why?
  - processing load distribution
  - load on the network backplane
Formal Specification of Data Aggregation

- Two basic operations
  - local processing: process the local dataset and generate a result
  - reduction operation: take two result datasets and generate one composite result dataset
- An data aggregation on N partitions includes N "local processing"s and N-1 "reduction operations".
- We assume the reduction operation is order-independent.

Application designers' responsibility
- examine application semantics and provide customized "local processing" and "reduction" routines

System support
- pretty much everything else

Flat Aggregation
- flat aggregation
- local processing at each partition
- send all results to the requesting client
- aggregate all results at the requesting client
- problem: the aggregation workload and incoming bandwidth at the client may become the bottleneck

Hierarchical Aggregation
- Question: how to form the hierarchy/tree?
- Targeting online systems
  - there are concurrent execution instances
  - performance metrics include the response time for individual requests and overall system throughput

Some Ideas
- Load-adaptive tree formation
  - nodes at different level of the reduction tree require different responsibility, e.g. leaf nodes do no reduction
  - low capacity or highly loaded nodes are placed in the tree such that they don't need to do much
- Deadline-driven scheduling
  - set different completion deadlines for nodes at different level of the tree in order to meet a particular overall completion time.
  - scheduling inside node is conducted according to deadlines of all local execution instances
- Others?
Tolerant of Partial Losses

- Many applications tolerate the loss of some partitions
  - keyword searching
  - satellite image composition
- What could be done in terms of runtime system support in this case?
  - figure out the performance metric
  - active pruning ...

Other Issues

- Programming Support
  - allows customized "local processing" and "reduction" routines be injected into all partition nodes
- Security
  - sandboxing user-injected routines, i.e., control their interaction with the runtime environment
  - e.g., allow only stream type I/O, disallow dynamic memory allocation