Achieving Data-Aware Load Balancing through Distributed Queues and Key/Value Stores

Ke Wang
Department of Computer Science
Illinois Institute of Technology
kwang22@hawk.iit.edu

Ioan Raicu
Department of Computer Science, Illinois Institute of Technology
Mathematics and Computer Science Division, Argonne National Laboratory
iraicu@cs.osu.edu

Abstract
Load balancing techniques (e.g., work stealing) are important to obtain the best performance for distributed task scheduling systems. In work stealing, tasks are randomly migrated from heavy-loaded schedulers to idle ones. However, for data-intensive applications where tasks are dependent and task execution involves processing large amount of data, migrating tasks blindly would compromise the data locality insuring significant data-transferring overhead. In this work, we propose a data-aware work stealing technique that combines key-value stores and distributed queues, which poses an enabling technique to achieve good load balancing, while maximizing data-locality. We leverage a distributed key-value store, ZHT, as a meta-data service that stores task dependency and data-locality information. We implement the proposed technique in MATRX, a distributed task execution fabric. We evaluate the work with all-pairs application structured as direct acyclic graph from biometrics, and compare with Falkon data-diffusion technique.

Contribution
1. Propose a data-aware work stealing technique that combines distributed queues and key-value stores.
2. Apply a distributed key-value store as a meta-data service to store important data dependency and locality information.
3. Evaluate the proposed technique up to hundreds of nodes showing good performance using different applications under different scheduling policies.

MATRX Architecture Overview

ZHT as Meta-Data Service

Distributed Queues

Decision Making Algorithm

Algorithm 1: Decision Making to Put a Task in the Right Ready Queue.

Input: a ready task (data), MTD (a threshold, t), current scheduler id (pid).

LReadyQ, SReadyQ, estimated length of the second (est_task_length) second.

Output: void.

if (pm all data_size <= est_task_length) then
  LReadyQ.push(task);
else
  long max_data_size = tm.data_size.at(0);
  int max_data_scheduler_idx = 0;
  for (i = 1 to tm.data_size.size(); do
    if tm.data_size.at(i) > max_data_size then
      max_data_size = tm.data_size.at(i);
      max_data_scheduler_idx = i;
    end
  end
  if (max_data_size <= est_task_length) then
    LReadyQ.push(task);
  else
    pm parental_list = pm.data_scheduler[0];
    pm.data_scheduler[0] = pm.data_scheduler[0] = id;
    LReadyQ.push(task);
  end
end

Applications

Image Stacking in Astronomy: the stacking of images from different parts of the sky. The stacking procedure involves the re-projecting each image to a common set of pixel planes, then co-adding many images to obtain a detection image that can attain a much higher signal-to-noise ratio.

Micro-Benchmarks

Conclusions

Applications for extreme-scales are becoming more data-intensive and fine-grained in both task size and duration. Task schedulers for data-intensive applications at extreme-scales need to be scalable to deliver the highest system utilization. Workload pose urgent demands for both load balancing and data-aware scheduling. This work combined distributed load balancing with data-aware scheduling through a data-aware work stealing technique. We implement the technique in MATRX, and apply a DKVX, as a transparent meta-data service. We evaluated our technique under four different scheduling policies with different workloads, and compared our technique with the Falkon data diffusion approach. Results showed that our technique is scalable to achieve both good load balancing and high location-hit rate. We have planned much work in the future, such as larger scales, HPC support, workflow integration, and MapReduce framework support.