On the Effectiveness of Application-Aware Self-Management for Scientific Discovery in Volunteer Computing Systems

Trilce Estrada¹, Michela Taufer²

¹University of New Mexico • ²University of Delaware

Contact: Trilce Estrada, estrada@cs.unm.edu

Overview

An important challenge faced by high-throughput, multi-scale applications is that human intervention has a central role in driving their success. However, manual intervention is inefficient, error-prone and promotes resource wasting. Additionally, scientific applications have diverse goals that are usually ignored by traditional self-management. We claim that to meet their scientific goals while using resources efficiently, applications require application aware-self-management. In this poster, we prove this claim for Volunteer Computing (VC) systems.

We present KOTree, a fully automatic method that organizes statistical information in a multidimensional structure that can be efficiently searched and updated at runtime. We integrate KOTree into a modular framework that provides application-aware self-management to VC systems. Our empirical evaluation shows that this framework meets application goals more effectively and uses resources more efficiently compared to other methods.

VC Parametric Applications

Volunteer Computing (VC) is a distributed paradigm where computers connected to Internet donate their idle cycles for scientific computation. Applications in VC systems are diverse in scope and objectives, making it harder to come up with a manual solution that can fit them all. The behavior of an application can be differently affected by one or more parameters. For example, in a Molecular Dynamics application, parameters can include initial velocity, solvent type, lattice granularity, and docking method. Different parameter values can affect accuracy of the solution, length of the simulation, and storage space, among many other variables.

We express applications as a sequence of parametric functions whose input, or domain, is given by the application parameters and the output, or codomain, is given by one application metric per function. We claim that high-throughput, multiscale applications need to become self-managed. An autonomic system, or self-managed system, has the capability of configuring, diagnosing, optimizing, and protecting itself. The idea is to:

- Use job parameters as coordinate in an N-dimensional space
- Use job results as they are collected to build a model that captures the impact of parameters on the application metrics
- Use the model to generate new jobs that are likely to advance application goals and use resources more efficiently

The goal of our application-aware self-management framework is to find parameter values within these ranges to optimize one or more metrics, meet the application goals, and use computing resources as efficiently as possible; all of that without direct human intervention.

**KOTree**

KOTree is an organized data structure that can predict multiple application metrics and can explore the N-dimensional parameter space effectively, while being built incrementally at runtime. Each node contains statistical information of jobs that are close in the parameter space.

We use the parameters of a job as coordinates to determine the path of the job in the KOTree. Given the initial space of parameters, we recursively divide this space into $2^N$ hypercubes of the same size. Every hypercube is given an unique identifier from 0 to $2^{N-1}$ based on its coordinates in the space.

KOTree is trained at runtime using metrics of collected jobs to update statistics of nodes in the tree. The training process starts with an empty KOTree and begins when we collect metrics for the first job. Each new job $x_i$ is associated to its metrics tuple, which is feed to the KOTree. To update statistics we use the Welford algorithm because it enables the accurate calculation of the running variance. When it is time to generate a new job, we use its path in a trained KOTree to predict its job metrics. For each node in the path, we keep track of its mean and variance. Then, we find the deepest node with lower variance that is larger than 0, and use its statistics of the metric of interest to make the prediction.

We keep a reduced list of nodes sorted based on their expected gain with respect to the application goal. For each node in the list, we generate jobs whose parameters fall within the specific hypercube ranges, proportionally to their relative gain with respect to the application goal. For each job, we generate its $N$ parameters using methods that promote exploration and exploitation of the parameter space.

We integrate KOTree into a modular framework to provide application-aware self-management to VC systems. This framework is composed by a learning engine and three modules: job generation, resource assessment, and result collection. The modules are used as software interfaces to provide a common communication protocol that can be understood by our KOTree regardless of the computational platform (e.g., VC, Grid, Cloud) and the actual scientific application it is optimizing.

### Evaluation

To evaluate our framework in a simulated VC environment, we define 3 case studies corresponding to types of applications with two metrics of interest and different goals:

- Minimizing expected CPU time per job
- Increasing accuracy of results regardless of the time it takes to compute them
- Finding a balance between faster jobs and accurate results

We evaluate the three case studies over a set of 14 scenarios corresponding to specific implementations of each application, with 1 to 4 parameters and compared our KOTrees (KOE and KOM) to a random (RND) and a simulated annealing (SAN) parameter exploration. KOTree outperforms a simulated annealing algorithm in 77% of the cases across application goals and application implementations. Additionally, KOE and KOM increase 85% of throughput in average. By building a statistical structure that embeds organized information, we can not only perform parameter exploration but also predict accurately job lengths, which is a crucial step towards efficient resource utilization.

**Acknowledgements.** This work was supported by the NSF IIS #0968350, SoCS - ExSciTecH: An Interactive, Easy-to-Use Volunteer Computing System to Explore Science, Technology, and Health.