Computer allocation for Kernel Execution using Artificial Intelligence

(CAKE using AI)

Ajay Ramesh & Chandra Kumar Basavaraj
ILLINOIS TECH

Abstract:

We are proposing a project where we can allocate computer or nodes for kernel function execution across heterogeneous systems using artificial intelligence. In section I, we are doing existing literature survey on computer for scheduling. Later In section II, we are giving enhanced ***proof of concept*** using AI to allocate computers. We also evaluate our model with existing model. The section III, we are writing research notes how to use artificial intelligence in Distributed Computing.

Background Information:

Often, we have heterogeneous systems, scheduling jobs is a problem in that, because it requires proper weighted allocation. For example if you have two computer with 8GB RAM, and 16GB RAM and you want to process some data. Allocating equal amount of data to those nodes is not a good idea. So we need to calculate properly on what machine can do before allocation. We see existing research promising enhanced artificial intelligence in scheduling will help the distributed computing smarter.

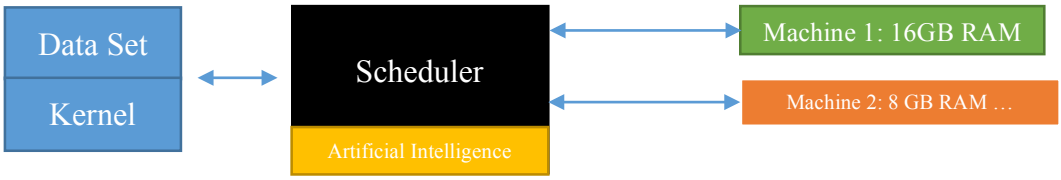

Fig 1 : Compute Allocation for kernel execution using AI.

Computers are evolving, that means a lot of heterogeneous system are coming up. For example at present let's say I have Ubuntu 16 GB but in future it will be outdated. That doesn't mean we can't use the Ubuntu for any sort of computation, we can use it but we need to understand what machine can do and can't. So, AI will help in uncovering all those aspects such that Computer allocation for kernel execution will simpler and smarted.

Goal :

Section I : Literature survey from top conference

In this phase we are going to do a thorough research existing scheduling algorithms, also present research from top reputed conferences. By doing so, we will come up where we need to improve and what techniques we can apply to make smart computer allocation for kernel function execution. After our survey complete we are hoping to make a blue print for the Proof of concept for section II, where we are going to demonstrate for a simple example.

Section II: Proof of concept and Evaluation

In this phase, we are developing scheduling algorithms, We have considered genetic algorithm and stimulated annealing to schedule. Based on new input from Section I, we might improve this section. (Fig 1 – gives the high-level overview of this proof of concept)

Abstract steps are –

1. Find a data set for searching (more than 1gb in size)
2. Write kernel function, that could be any search algorithm/any algorithms. Ex – Binary Search algorithm.
3. Choose EC2 instances, note to keep all instances heterogeneous. So that no EC2 instance are same.
4. Now apply Genetic algorithm/Stimulated annealing to come with *scheduling plan*.
5. As per the ***plan*** from step 4 we are going to upload the part of dataset, and kernel function to particular EC2 instance. Trigger start on those nodes for execution.
6. Collect the search results.

Evaluation –

1. We will be going to check the performance of traditional scheduling algorithm and other methods(any new methods we sought in section I) for the same data set.

Section III: Research Notes from present research and our advice on those.

- Exploration on application of Deep Neural Network.
- Notes on how to secure application using AI.
- Notes on Application of control theory.
- Notes on how we can use Natural language processing in Computer allocation for UI.

Execution Plan:

SL No	Project Phase	Start Date	End Date	Duration
1	Literature Survey (Gathering required information)	2/13/2017	3/5/2017	3 weeks
2	Design, Review and Implementation	3/6/2017	3/27/2017	3 weeks
3	Testing and Verification against goals	3/28/2017	4/7/2017	1.5 weeks
4	Project presentation	4/8/2017	4/27/2017	3 weeks
5	Final report submission	4/27/2017	4/30/2017	0.5 weeks

Reference –

1. SOLVING BIG QUESTIONS REQUIRES BIG COMPUTATION //news.stanford.edu/features/2014/computing/
2. "The Evolution of Cluster Scheduler Architectures", http://firmament.io/blog/scheduler-architectures.html
3. "Job Scheduling in HPC Clusters",http://www.dell.com/downloads/global/power/ps1q05-20040135-fang.pdf
4. M. Nikravan and M. Kashani, "A genetic algorithm for process scheduling in distributed operating systems considering load balancing", 21st European Conference on Modeling and Simulation, June 2007.
5. L. Schmitt, "Fundamental Study Theory of Genetic Algorithms", International Journal of Modelling and Simulation Theoretical Computer Science, vol. 259, May 2001, 1-61.
6. A. Omara and M. Arafa, "Genetic Algorithms for Task Scheduling Problem", Studies in Computational Intelligence, Springer Berlin, Heidelberg, Germany, vol. 203, May 2009, 479-507.
7. F. Lin and C. Hsu, "Task assignment scheduling by simulated annealing", IEEE Region 10 Conference on Computer and Communication Systems, vol. 1, September 1990, 279-283.
8. S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by Simulated Annealing", Science, vol. 220, May 1983, 671-680.
9. A. Haghighat and M. Nikravan, "A Hybrid Genetic Algorithm for Process Scheduling in Distributed Operating Systems Considering Load Balancing", Parallel and Distributed Computing and Networks, February 2005.
10. Apache Spark Research - Discretized Streams: Fault-Tolerant Streaming Computation at Scale - Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, Ion Stoica, University of California, Berkeley
11. Apahce Spark Research - http://spark.apache.org/research.html