Cost-Time Performance of Scaling Applications on the Cloud

Sunimal Rathnayake  
Department of Computer Science  
National University of Singapore  
Singapore  
sunimalr@comp.nus.edu.sg

Lavanya Ramapantulu  
Computer Systems Group  
International Institute of Information Technology  
Hyderabad, India  
lavanya.r@iiit.ac.in

Yong Meng Teo  
Department of Computer Science  
National University of Singapore  
Singapore  
teoym@comp.nus.edu.sg

Abstract—Recent advancements in big data processing and machine learning, among others, increase the resource demand for running applications with larger problem sizes. Elastic cloud computing resources with pay-per-use pricing offers new opportunities where large application execution is constrained only by the cost budget. Given a cost budget and a time deadline, this paper introduces a measurement-driven analytical modeling approach to determine the largest Pareto-optimal problem size and its corresponding cloud configuration for execution. We evaluate our approach with a set of representative applications that exhibit a range of resource demand growth patterns on Amazon AWS cloud. We show the existence of cost-time-size Pareto-frontier with multiple sweet spots meeting user constraints. To characterize the cost-performance of cloud resources, we use Performance Cost Ratio (PCR) metric. We extend Gustafson’s fixed-time scaling in the context of cloud, and, investigate fixed-cost-time scaling of applications and show that using resources with higher PCR yields better cost-time performance. We discuss a number of useful insights on the trade-off between the execution time and the largest Pareto-optimal problem size, and, show that time deadline could be tightened for a proportionately much smaller reduction of problem size.

Index Terms—scaling, largest problem size, cloud, cost-time performance, Pareto-optimal configuration

I. INTRODUCTION

Scalability in cloud computing is of two-fold: application scalability and resource scalability. Variation of resource demand of application with problem size is known as application scalability. The plethora of cloud resources available on cloud with different cost and performance leads to resource scalability. Unlike in traditional on-premise computer systems where scalability was limited by the available on-premise hardware resources, scalability on cloud is bounded only by cost budget and execution time deadline. As cloud resources are characterized in terms of processing power, memory, storage, and, network performance, among others, the cloud consumer has the opportunity to choose a cost-efficient cloud resource configuration to match the resource demand of the application. However, the consumer should take into account both the resource demand growth of the application as well as the cost-performance of cloud resources when deciding on an optimal cloud resource configuration.

Scalability in parallel computing has been well studied [7], [9], [20] with a majority focusing on time-performance. Gustafson’s law on fixed-time scaling is one such work which argues that given a fixed-time, a near-linear parallel speedup could be achieved when the size of the application grows [4]. With heterogeneous scalable cloud resources and application scalability, we extend Gustafson’s law and investigate fixed-cost-time scaling of applications on the cloud. Moreover, since cloud resource usage is measured, metered, and, charged based on the duration the resource was used, studying the impact of scaling on cost is equally important.

Given an application with a time deadline and a cost budget, we propose a measurement-driven analytical modeling approach to determine cost-time Pareto-optimal problem sizes (defined in Section III-C) and cloud resource configurations for executing them. Although Pareto-optimal scaling is not new to parallel computing [2], [17], applying Pareto-optimal scaling for investigating the trade-off between application problem size, and, the cost and time of execution is new in the context of cloud computing. Similar to Hwang et al.’s [10] work on performance evaluation, we use the instruction count as the proxy for matching the resource demand of application with resource capacity of cloud resources. We utilize baseline executions for characterizing applications and resources, and, evaluate our approach on Amazon EC2 cloud.

II. RELATED WORK

Starting from the early research such as Amdahl’s law and Gustafson’s law, researchers [7], [9], [10], [20], [21] have attempted to model and improve scalability. In cloud computing, most of the work focus on resource elasticity [1], [3], [8], [11], [13]–[15], [19]. Such approaches include autoscaling and resource migration for cost optimization, scheduling for dynamic pricing schemes such as AWS spot pricing, and, model driven approaches for predicting resource demand. In comparison, fewer research works [5], [6], [18], [22], [23] do address application scalability on the cloud. These research works include resource optimization for elastic applications, benchmarking for determining the best virtual machine type, and, framework to develop elastic algorithms. Extending the current state of research, this paper focuses on fixed-cost-time Pareto-optimal scaling of application problem size using a measurement driven analytical modeling approach.

III. APPROACH

This section discusses our measurement-driven analytical
model derivation and Pareto optimization algorithm.

A. Fixed Cost-time Application Scaling

As shown in Figure 1, given an application \( P \) with a cost budget \( C \), a time deadline \( T \), and, a set of cloud resources, our approach determines Pareto-optimal problem sizes \( S \) of \( P \) executable on the cloud and Pareto-optimal cloud resource configurations to execute them.

For application characterization, we measure the number instructions executed on the non-virtualized server while running the application for different problem sizes. These measurements are utilized to derive the application resource demand growth function using a curve fitting technique. For characterizing cloud resources, the execution rate for each cloud resource instance is computed by dividing the number of instructions measured on a non-virtualized server for each problem size by the corresponding execution time.

Our Pareto-optimization algorithm takes as input \( C, T, \) the set of cloud configurations \( G \) and the growth function of \( P \), \( f(S) \), and determines largest Pareto-optimal sizes of \( P, S^\text{max} \), optimal cost \( C^* \), optimal time \( T^* \) and the corresponding cloud resource configuration \( \text{config} \).

B. Model Derivation

A cloud configuration is a combination of instances from one or more cloud resource types. We use the notation \( \{r_{0,j}, r_{1,j}, \ldots, r_{i,j}, \ldots, r_{|R|-1,j}\} \) to denote \( G_j \), the \( j \)th cloud configuration in \( G \), where \( r_{i,j} \in R \) is the set of cloud resource instances from resource type \( i \in R \).

The largest \( S \) of \( P \) executable on \( G_j \) depends on the number of instructions \( G_j \) can execute while running \( P \),

\[
S = f^{-1}(W) \tag{1}
\]

where \( f^{-1} \) is the inverse of the scaling function of \( P \), and, \( W \) is the total number of instructions executed on \( G_j \).

Suppose, \( G_j \) has to execute \( P \) for a time duration \( t \) to execute \( W \) instructions, thus, \( W \) is computed as

\[
W = \Delta_{G_j} \times t = \sum_{i=0}^{\infty} (|r_{i,j}| \times \Delta_i) \tag{2}
\]

where \( \Delta_{G_j} \) is the resource capacity of \( G_j \).

Resource capacity of a cloud resource configuration is defined as the summation of resource capacities of all cloud resource instances in the configuration.

\[
\Delta_{G_j} = \sum_{i=0}^{\infty} (|r_{i,j}| \times \Delta_i) \tag{3}
\]

where \( |r_{i,j}| \) is the number of cloud resource instances from resource type \( i \) in \( G_j \), and, \( \Delta_i \) is the resource capacity of cloud resource type \( i \).

A cloud resource instance consists of one or more vCPUs, thus, the total resource capacity of a cloud resource instance depends on the number of vCPUs it consists.

\[
\Delta_i = \Delta_{\text{vCPU}} \times v_i \tag{4}
\]

where \( \Delta_{\text{vCPU}} \) is the per-vCPU resource capacity and, \( v_i \) is the number of vCPUs in one instance of \( i \).

Given that it takes time \( t \) to execute size \( S \) of \( P \) on configuration \( G_j \), the total cost is

\[
C' = t \times C_{G_j} \tag{5}
\]

where \( C_{G_j} \) is the cost per unit time for configuration \( G_j \). \( C_{G_j} \) is determined based on the cost for unit time of each cloud resource instance in the \( G_j \) and is defined as

\[
C_{G_j} = \sum_{i=0}^{\infty} (|r_{i,j}| \times c_i) \tag{6}
\]

where \( c_i \) is the cost per unit time for cloud resource type \( i \).

C. Determining Largest Pareto-optimal Size

We need to maximize the size while minimizing cost and time. Thus, we define that a tuple \( <\text{size}, \text{cost}, \text{time}> \) consists Pareto-optimal size if size cannot be increased without increasing either/both cost and time. Our Pareto-optimization algorithm traverses all cloud configurations and builds a list of tuples of which each tuple contains maximum problem size for each configuration, cost, and, time. Secondly, tuples with same time are clustered together. Thirdly, from each cluster with same time, the tuples with minimum cost are retained and others are discarded. From the remainder, the tuples with largest size are selected as Pareto-optimal sizes.

IV. EVALUATION

This section presents the evaluation of our approach Amazon EC2 cloud with a representative set of applications followed by an analysis of important observations.

A. Applications

To evaluate our approach we selected two applications that exhibit linear, quadratic and logarithmic growth of resource demand with respect to problem size as shown in Table II. \( n\)-body \( (n, s) \) [12] implements \( n\)-body simulation of masses on MPI where scalable problem sizes are the number of masses \( n \) and number of simulation steps \( s \). sand \( (n, \tau) \) [16] is a genome sequence alignment application implemented on Work-Queue platform which aligns compatible genome sequences from a list of candidate sequences of size \( n \) based on a quality threshold \( \tau \).

<table>
<thead>
<tr>
<th>Application</th>
<th>Scaling Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>n-body</td>
<td>( W = 10^n n^2 + 5 \times 10^n n - 7 \times 10^n )</td>
</tr>
<tr>
<td>sandy</td>
<td>( W = 10^n s^2 + 2 \times 10^n s )</td>
</tr>
<tr>
<td></td>
<td>( W = 3 \times 10^n \text{vCPU} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>Scaling Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>n-body</td>
<td>( W = 10^n n^2 + 5 \times 10^n n - 7 \times 10^n )</td>
</tr>
<tr>
<td>sandy</td>
<td>( W = 10^n s^2 + 2 \times 10^n s )</td>
</tr>
<tr>
<td></td>
<td>( W = 3 \times 10^n \text{vCPU} )</td>
</tr>
</tbody>
</table>
Fig. 2: Performance Cost Ratio of EC2 Resources Types

Table III: Amazon EC2 Cloud Resource Types

<table>
<thead>
<tr>
<th>Resource</th>
<th>vCPUs</th>
<th>Freq. (GHz)</th>
<th>Mem. (GB)</th>
<th>Cost ($)</th>
<th>Execution Rate (billion instructions/sec/vCPU)</th>
<th>PCR (billion instructions per $)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c4.large</td>
<td>2</td>
<td>2.9</td>
<td>3.75</td>
<td>0.105</td>
<td>1.38</td>
<td>54.61</td>
</tr>
<tr>
<td>c4.xlarge</td>
<td>4</td>
<td>2.5</td>
<td>8.3</td>
<td>0.82</td>
<td>3.94</td>
<td>99.99</td>
</tr>
<tr>
<td>n4.large</td>
<td>2</td>
<td>2.5</td>
<td>6</td>
<td>0.133</td>
<td>11.66</td>
<td>99.99</td>
</tr>
<tr>
<td>n4.xlarge</td>
<td>4</td>
<td>2.5</td>
<td>16</td>
<td>0.268</td>
<td>8.85</td>
<td>99.99</td>
</tr>
<tr>
<td>m4.large</td>
<td>2</td>
<td>2.5</td>
<td>6</td>
<td>0.268</td>
<td>8.85</td>
<td>99.99</td>
</tr>
<tr>
<td>m4.xlarge</td>
<td>4</td>
<td>2.5</td>
<td>16</td>
<td>0.268</td>
<td>8.85</td>
<td>99.99</td>
</tr>
<tr>
<td>r3.large</td>
<td>2</td>
<td>2.5</td>
<td>15</td>
<td>0.166</td>
<td>3.69</td>
<td>49500</td>
</tr>
<tr>
<td>r3.xlarge</td>
<td>4</td>
<td>2.5</td>
<td>40.5</td>
<td>0.325</td>
<td>11.44</td>
<td>4B07141</td>
</tr>
</tbody>
</table>

B. Cloud Resources and Performance Cost Ratio

We use Performance Cost Ratio (PCR) as a metric for representing and comparing cost efficiency of cloud resources. PCR is defined as

\[
PCR = \frac{\text{Instruction Execution Rate}}{\text{Cost}} \quad \text{(instructions/sec)/(cost/$)}
\]

Figure 2 shows PCR of four non-accelerated resource categories on Amazon EC2 cloud for n-body and sand applications and we observe that PCR has a non-linear relationship across different resource categories. For evaluation of our approach, while preserving the non-linear relation in PCR across resource categories, we selected c4, m4 and r3 resource categories as shown in Table III. From each category, we selected large, xlarge, 2xlarge resource types resulting in a configuration space of over ten million configurations.

C. Validation

Table IV shows model validation where the predicted time and cost are compared against measured values from Amazon EC2 cloud. Due to limited research budget, we validate only the minimum and maximum problem sizes for each application shown in Figure 3. The prediction accuracy varies in the range 81% - 82% and 84% - 87%, for n-body and sand respectively. Sources of inaccuracy include impact of virtualization on measurements, having different system specification for the same resource type on cloud, and the communication overhead.

D. Model Analysis

This section presents an analysis based on model predictions for scaling applications on the Amazon EC2 cloud. For experiments on cloud, for simplicity and due to research budget constraints, we set the time deadline to 24 hrs and cost budget to $100 whereas for extrapolated analysis in sections IV-D2 and IV-D3 we set time deadline to 7 days (168 hrs) and cost budget to $1000, hence emulating long-running applications.

1) Pareto-optimal Problem Size

To understand the relationship between the application problem size and the cost-time performance of cloud resources, as shown in Figure 3, we investigate how Pareto-optimal problem sizes are distributed in the cost-time-size space.

Observation 1: Given an application with a cost budget and time deadline there exist one or more Pareto-optimal problem sizes of which one consists of the largest problem size. Having a set of Pareto-optimal problem sizes within the predefined cost and time constraints provides opportunity for cloud consumer to further tighten these constraints at the expense of the problem size of the application.

Observation 2: Increasing the cost budget and relaxing the time deadline does not always allow a larger problem size. Blindly selecting the cloud resource configuration may result in a smaller problem size even with larger execution time and higher cost, thus, consideration of cost-performance of different resource combinations is important.

Observation 3: For a given Pareto-optimal problem size with multiple resource configurations, resource demand is allocated to different resource types in order of higher PCR. Selecting the resources in the order of the highest PCR would obtain a configuration with a near-optimal problem size.

2) Impact of Time Deadline on Scaling

To investigate the impact of time deadline on scaling, as shown in Figure 4, we determine the largest size of the application executable for n-body and sand while relaxing the time deadline for different fixed cost values.

Observation 4: Among Pareto-optimal problem sizes, tightening of time deadline results in a proportionately smaller reduction of largest problem size. When the time deadline is relaxed, the capability of resources increase linearly, but the resource demand increases quadratically. Hence, results in a sublinear growth of largest problem...
scaling decisions on cloud is challenging due to the large configuration space and application dependent resource demand growth. This paper presents a measurement-driven analytical modeling approach to determine the largest Pareto-optimal problem sizes for a given application with a time deadline and a cost budget, and, cloud configurations to execute them. Using model results, we show the existence of cost-time-size Pareto frontier with multiple Pareto-optimal sizes for a given application with a time deadline and cost budget. We present interesting insights on fixed-cost-time scaling on cloud and introduce Performance Cost Ratio (PCR) which can be utilized to determine near-optimal cloud resource configurations.

VI. ACKNOWLEDGMENTS

This work was supported by Singapore Ministry of Education through the Academic Research Fund Tier 1. The authors thank Amazon Web Services for AWS Cloud research credits.

REFERENCES