Minimizing Computing Core Costs in Cloud Infrastructures that Host Location-Based Advertising Services

Vikram K. Ramanna, Christopher P. Paolini, Mahasweta Sarkar, *Member, IAENG*, and Santosh Nagaraj

***Abstract*— Cloud computing provides services to a large number of remote users with diverse requirements, an increasingly popular paradigm for accessing computing resources over the Internet. A popular cloud-service model is Infrastructure as a Service (IaaS), exemplified by Amazon's Elastic Computing Cloud (EC2). In this model, users are given access to virtual machines on which they can install and run arbitrary applications, including relational database systems and geographic information systems (GIS). Location-based services (LBS) for offering targeted, real-time advertising is an emerging retail practice wherein a mobile user receives offers for goods and services through a smart phone application. These advertisements can be targeted to individual potential customers by correlating a smart phone user's interests to goods and services being offered within close proximity of the user. In this paper, we examine the problem of determining the appropriate number of microprocessor cores assigned to a relational database instance (virtual machine) managed by a cloud infrastructure, required to constrain the query response time for a targeted advertisement to reach a mobile customer within approachable distance to a Point of Sale (POS). We assume the optimum number of cores required to minimize offer latency is one that minimizes microprocessor core expenses, charged by cloud infrastructure providers, while maximizing application service provider revenues derived from POS transaction fees. Changes in the number of microprocessor cores assigned to database resources can result in changes in the time taken to transmit, receive, and interpret a targeted advertisement sent to a potential customer in motion. We develop a methodology to establish an equilibrium state between the utility gained from POS transaction revenues and costs incurred from purchasing microprocessor cores from infrastructure providers. We present different approaches based on an exponential and linear method to model customer purchase decisions. From these models, the marginal cost and marginal revenue is calculated to determine the optimal number of microprocessor cores to purchase and assign to database instances executing within cloud infrastructures.**

***Index Terms*— Location-based service, Cloud computing, Mobile computing, Mobile advertising, IaaS**

Manuscript received July 16, 2013; revised August 16, 2013.

V. K. Ramanna is with San Diego State University, San Diego, CA 92182 USA (phone: 619-594-7159; fax: 619-594-2068; e-mail: vikram1986@gmail.com).

C. P. Paolini is with San Diego State University, San Diego, CA 92182 USA (e-mail: paolini@engineering.sdsu.edu).

M. Sarkar is with San Diego State University, San Diego, CA 92182 USA (e-mail: msarkar2@mail.sdsu.edu).

S. Nagaraj is with San Diego State University, San Diego, CA 92182 USA (e-mail: snagaraj@mail.sdsu.edu).

I. Introduction

Location-based services refer to computational services that locate a mobile user geographically and deliver information to the user tailored to their location. A *Location Based Service* (LBS), specifically, is a geospatial mobile-based application that provides services based on a user's geographical location. An example of such a service is a food service application that informs mobile device holders of nearby restaurants. First generation LBSs were reactive and client-server focused where users would query for information and received a response back from the server. With the advancement of push notification techniques, improved mobile Internet access, and widespread adoption of the Web2.0 paradigm, next generation user-to-user interactive LBSs evolved where information is pushed asynchronously to users based on their location, rather than users having to explicitly query for information.

Location-based advertising services refer to marketing services offered to smart-phone users within the proximity of a point of sale (POS). Suppose a smart-phone application firm wishes to host a LBS in a cloud infrastructure. An LBS will rely on a relational database to monitor, in real-time, the geospatial data of all application users. The firm must first decide which particular cloud infrastructure provider to invest in, such as Amazon EC2, Microsoft Windows Azure, Rackspace, etc. Once a firm has chosen an infrastructure provider, the firm deploys their database in a cloud virtual machine (VM) instance. The firm's database is used to maintain the purchasing preference profiles of application users in addition to geospatial information. As mobile users enter within a proximate distance or *purchase frontier* of a POS, the deployed database is used to send offers for goods or services to application users. Whenever a mobile user receives an offer on his or her smart-phone, the user has a probability of honoring the offer and changing course to walk to the POS to make a purchase. Each purchase instigated from a received offer will generate transaction fees returned to the application firm in the form of revenue. The firm is thus faced with a business optimization problem: whether to add or subtract additional cloud infrastructure resources to maximize profit. Adding additional resources can reduce *offer latency*, or the time between when a user enters the purchase frontier of

ISBN: 978-988-19252-3-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

a POS and the time when the user receives an offer on his or her smart-phone. However, adding additional resources incurs additional costs for the firm. The firm must decide the optimal number of resources to purchase from the cloud provider to maximize profit.

Cloud computing involves delivery of hosted services over the Internet or a Local Area Network (LAN) [1]. These services can be broadly classified into three categories: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) and *Software as a Service* (SaaS). Infrastructure as a Service is a provision model in which an organization outsources resources such as hardware, storage, servers, and networking components to support operations on demand. The service provider is completely responsible for running, housing, and maintaining these resources, which are distributed as a service and dynamically scaled according to demand. The IaaS client typically is billed on a per-use basis. The IaaS model is suitable where the demand is very volatile, that is, where there are spikes or troughs in terms of demand of infrastructure and where there is a limit to capital expenditure for a specific business application.

The problem addressed in this paper is to derive a way for cloud customers to choose, dynamically, an optimal number of microprocessor cores to drive a relational database, hosted in a cloud infrastructure, that supports location based advertising services for mobile smart-phone applications. Upon choosing a particular commercial cloud provider, a mobile application firm hosts their back-end database application within a cloud instance, which is used to send offers for good or services to mobile smart-phone users. Based on the revenue generated by transaction fees charged to smart-phone users who consent to an offer received by purchasing goods or services at a point of sale, the mobile application firm then chooses to either add or remove microprocessor cores to maximize revenue.

II. Modeling Mobile Shoppers

Fig. 1 shows an aerial photograph of an outdoor shopping mall where customers walk within regions bounded by brick-and-mortar stores. Suppose a mobile shopper is walking along the yellow path as shown. A store, located at the black dot labeled "POS", transmits offers to customers as soon as they enter the 50m purchase frontier, identified by the outer concentric red circle. Circles are shown in increments of 10m from the POS. The greater the offer latency, the greater the distance traveled by a shopper and the greater the shopper will need to backtrack to the POS to honor the received offer and make a purchase. Based on the number of microprocessor cores assigned to the VM that hosts the geospatial database, the shopper will experience different offer latencies, which will result in the shopper receiving an offer at different distances from the POS. Suppose many cores were assigned to the VM. Then, the offer latency would be shorter and the shopper might receive an offer at position D_1. However, suppose fewer cores were assigned to the VM. In this case, the shopper would travel an additional distance, say D_3, before receiving the offer. At position D_1, the user will need to *backtrack* a shorter distance to the POS than at position D_3.

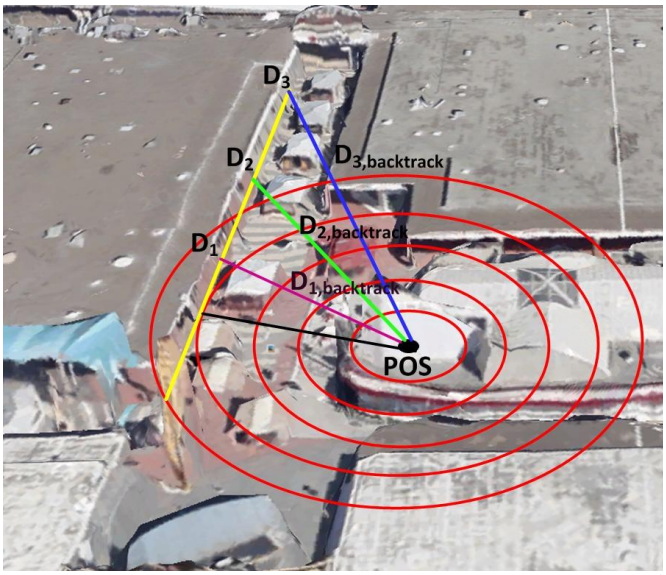

Fig. 1. Typical location-based customer purchase scenario. Aerial view of an outdoor shopping mall where a customer walks along the yellow path shown. A store, located at the black dot labeled POS, transmits offers to customers as soon as they enter the 50m purchase frontier, identified by the outer concentric red circle. Circles are shown in increments of 10m from the POS. The greater the offer latency, the greater the distance traveled by a shopper and the greater the shopper will need to backtrack to the POS to honor the received offer and make a purchase.

Our customer purchase model assumes the probability a user will backtrack to a given POS is indirectly proportional to the distance the user is to the POS at the time of receiving an offer. Thus, a shopper who receives an offer at D_1 is more likely to walk to the POS than a shopper who receives an offer at D_3. By increasing the number of cores assigned to the database VM, a firm can reduce the shopper's backtrack distance by reducing the offer latency. However, adding cores incurs additional cost for the firm, according to the cloud provider's resource pricing model. To maximize profit, the mobile application firm must minimize both a mobile user's backtrack distance by minimizing offer latency, and minimize the cost of adding additional cores assigned to a database's virtual instance.

To simulate offer latency as a function of core count, we constructed a private cloud using the open source Eucalyptus Cloud platform [2] used to build Amazon Web Services (AWS)-compatible private and hybrid clouds. Eucalyptus is a private cloud-computing platform that implements the Amazon specification for EC2, S3, EBS, and IAM. We developed a multithreaded code that simulates one million mobile LBS users by simultaneously uploading one million random geospatial records into a remote PostgreSQL database running within a Eucalyptus virtual machine instance. Each geospatial record is 0.5MB in length and stores a user's GPS data (latitude, longitude, altitude, accuracy, GPS fix time) and consumer interest data. The total time required to upload one million geospatial records is given by

$$t_{\text{Total}} = t_{\text{Insertion}} + t_{\text{Geospatial}} + t_{\text{Corelation}} + t_{\text{Reception}} \quad (1)$$

where $t_{Insertion}$ is the time required to transmit and insert records into a PostgreSQL table, $t_{Geospatial}$ is the time required

ISBN: 978-988-19252-3-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

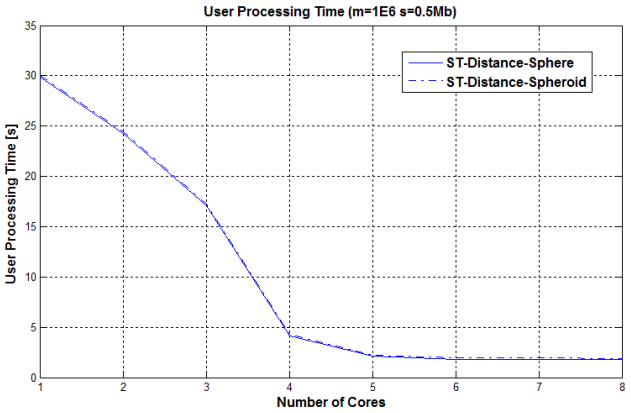

Fig. 2. Per user geospatial processing time t_{User} as a function of microprocessor core assignment. Assumes one million mobile users and a geospatial update message size of 0.5 MB. Processing time is invariant with five or more cores. ST_Distance_Sphere uses the haversine formula to compute the distance between each user and the POS; ST_Distance_Spheroid() uses Vincenty's method.

to compute the distance between the user and the POS using either the haversine formula [3] or Vincenty's method [4], $t_{Corelation}$ is the time required to compute a correlation coefficient between the user's commercial interests and the products and/or services being offered by the POS, and $t_{Reception}$ is time required to receive an offer from the cloud infrastructure, assuming a 4G mobile network is being used (8 Mbps download data rate, 4 Mbps upload data rate). Fig. 2Fig. shows a plot of the per user geospatial processing time t_{User} as a function of microprocessor core assignment, where $t_{User} = t_{Total}/N$ with N = 1 million. One can see that the processing time to receive an offer - or offer latency - is reduced as additional cores are attached to the database VM instance, up to 5 cores when the processing time becomes invariant.

Assuming an average walking speed of v = 5 km/h (1.4 m/s)

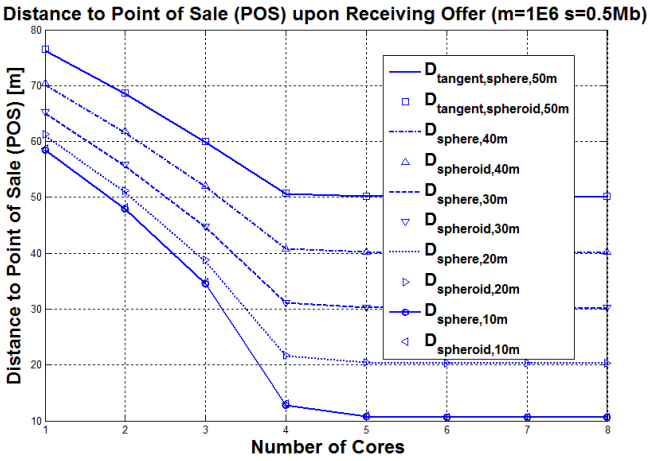

Fig. 3. Shopper backtrack distance as a function of database VM microprocessor core count, for five radii at 50, 40, 30, 20, and 10 meters from the POS.

[5], we can compute the backtrack distance, $d_{Backtrack}$, simply as $d_{Backtrack} = \sqrt{d^2 + r^2}$ where $d = v\, t_{User}$ is the distance traveled by the shopper and r is the radial distance the user is from the POS, assuming the user tangentially crosses a given concentric circle within the purchase frontier. Fig. 3 shows a shopper's required backtrack distance in meters as a function of core count, for five radii at 50, 40, 30, 20, and 10 meters from the POS. After five or more cores are assigned to a VM, the backtrack distance becomes invariant. We assume the reluctance of a shopper to travel the backtrack distance to a POS is proportional to the distance. We modeled shopper

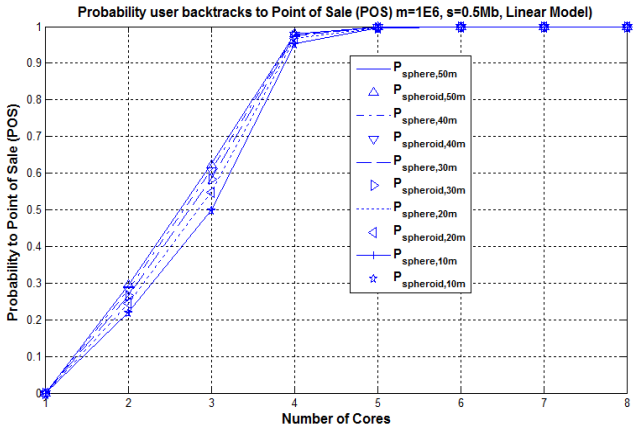

Fig. 4. Probability a shopper will backtrack to a POS upon receiving an offer as a function of microprocessor core count, based on a linear probability model, for five radii at 50, 40, 30, 20, and 10 meters from the POS.

reluctance or purchase probability using a linear and exponential model.

The simplest approach to modeling purchase probability uses a linear model where the probability a shopper backtracks to a point of sale upon receiving offer on a 4G mobile smart phone is given by

$$P_{\text{linear}} = \frac{d_{backtrack} - d_{\max}}{d_{\min} - d_{\max}} \qquad (2)$$

where d_{max}, d_{min} are the maximum and minimum distances a shopper could have travelled upon receiving an offer for goods and/or services. From Fig. 4, based on a linear model, a shopper is guaranteed to make a purchase if the LBS database VM is assigned five or more cores. We hypothesized that shopper reluctance is more realistically modeled using an exponential where reluctance to backtrack increases exponentially with backtrack distance, that is

$$P_{\text{exponential}} = \lambda e^{-\lambda\beta x} \qquad (3)$$

with λ=1 and scaling factor β=1/50. Fig 5 shows shopper purchase probability as a function of core count based on an exponential model. The scaling factor adjusts the purchase probability to fall in the 20% to 80% range, based on the number of assigned cores.

ISBN: 978-988-19252-3-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

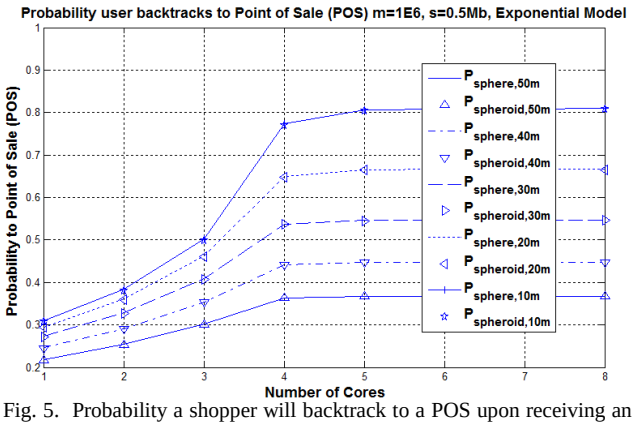

Fig. 5. Probability a shopper will backtrack to a POS upon receiving an offer as a function of microprocessor core count, based on an exponential probability model, for five radii at 50, 40, 30, 20, and 10 meters from the POS.

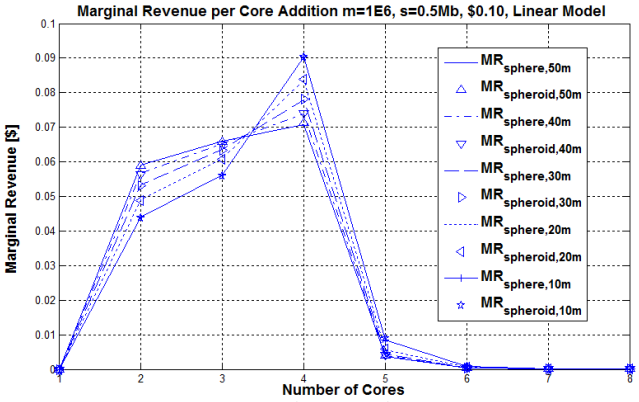

Fig. 6. Marginal revenue as a function of microprocessor core count, based on a transaction fee of 10¢ per purchase and a linear probability model, for five radii at 50, 40, 30, 20, and 10 meters from the POS.

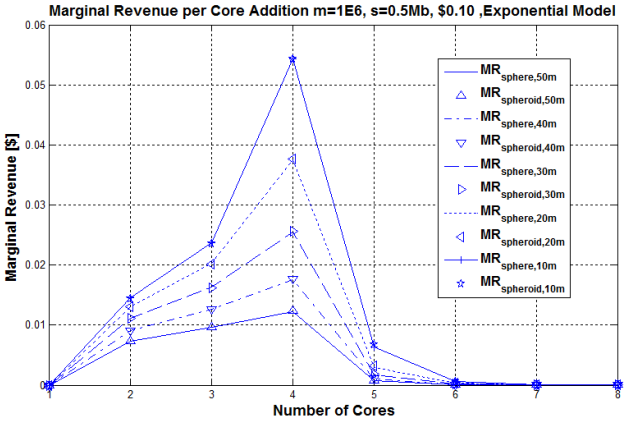

Fig. 7. Marginal revenue as a function of microprocessor core count, based on a transaction fee of 10¢ per purchase and an exponential probability model, for five radii at 50, 40, 30, 20, and 10 meters from the POS.

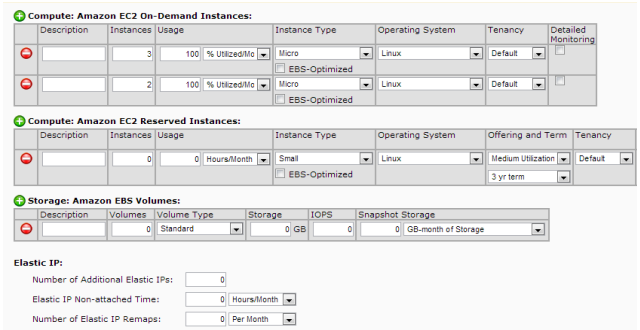

Fig. 8. Marginal cost can be determined using the Amazon EC2 online calculator, based on a firm's chosen instance type.

III. Revenue and Cost

A mobile application firm's profit will be maximum when marginal revenue equals marginal cost. Marginal revenue (*MR*) is the additional revenue obtained by the mobile application firm via transaction fees collected from purchases when one additional microprocessor core is assigned to the database virtual instance. Marginal cost (*MC*) is the additional cost incurred by the mobile application firm from purchasing one additional microprocessor core from the cloud infrastructure provider. When *MR>MC*, the firm can increase profit by adding microprocessor cores to the VM. When *MR<MC*, the firm can increase profit by revoking microprocessor cores assigned to the database VM, where the revoked cores are then returned to the cloud for use by a different client. It is in the firm's best interest to assign the particular number of cores to the VM such that the equilibrium condition *MR=MC* is satisfied. We calculate the revenue *R* on each purchase as the product of the purchase transaction fee *T* and purchase probability *P*,

$$R = PT \tag{4}$$

and marginal revenue as

$$MR = \frac{\Delta R}{\Delta c} \tag{5}$$

with $\Delta c=1$ for the increase in revenue per each one additional microprocessor core added to the VM. Figures 6 and 7 show marginal revenue as a function of microprocessor core count, based on a transaction fee of 10¢ per purchase for five radii at 50, 40, 30, 20, and 10 meters from the POS, for the linear and exponential models, respectively. Marginal cost is calculated as

$$MC = \frac{\Delta C}{\Delta c} \tag{6}$$

for the increase in cost the application firm must pay the cloud provider per each one additional microprocessor core added to the VM. Based on a chosen instance type [6] (e.g. micro, small, medium, etc.) which determines an initial resource configuration, one can compute marginal cost using a provider's online calculator [7], as shown in Fig. 8. For example, Amazon EC2 charges hourly usage fees for an on-demand instance. Pricing is based on each instance-hour consumed for each VM instance purchased by the mobile application firm. The marginal cost calculations used in this paper are based on an EC2 *micro* instance, which provides a small amount of consistent CPU resources to an application and allows one to increase CPU capacity in short bursts when

ISBN: 978-988-19252-3-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

additional CPU cycles are needed.

IV. Results

Selected marginal revenue and cost plots are shown in Figs. 9-11 for different transaction fees and purchase probability models. The plots show that, for a particular transaction fee and purchase probability model, the profit maximizing number of microprocessor cores to assign a virtual instance running a database can be determined by locating the integer number of cores closest to the average abscissa of the points of intersection. In Fig. 9, the optimal number of cores to assign is three, while in Fig. 10 and 11 the optimal number is five.

V. Conclusion

We identified that a firm can use the *MR=MC* profit maximization concept to determine the optimal number of compute cores to assign to a database virtual instance hosted in a cloud infrastructure. This is a new and simple strategy that firms can use when hosting location-based mobile smart-phone applications in a commercial cloud infrastructure, for apps that provide advertising services. While the results are based on the Amazon Elastic Compute Cloud (Amazon EC2) infrastructure, the methodology can be adapted to any commercial cloud provider, where revenue is earned through application usage and costs are incurred through cloud resource investment. Furthermore, more complex purchase probability models can be substituted in place of our simplified linear and exponential models. Our purchase probability models are purely hypothetical and need to be explored and validated as a future effort, by collecting data on actual customer behavior after receiving offers for goods and services by nearby points of sale.

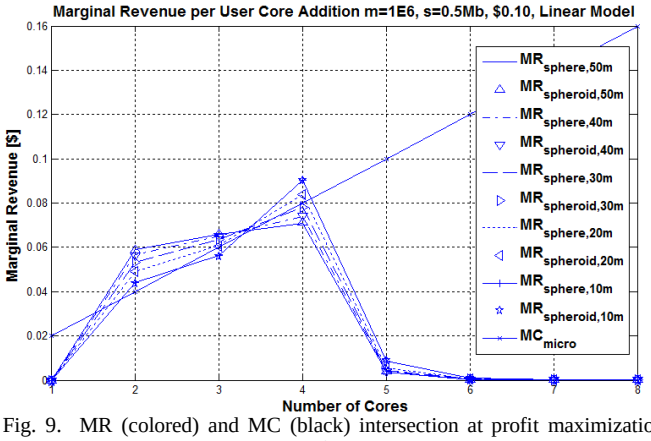

Fig. 9. MR (colored) and MC (black) intersection at profit maximization points with respect to the number of cores assigned to the database virtual instance. Linear purchase probability model and transaction fee of 10¢ per purchase. Optimal number of cores to assign the instance is three.

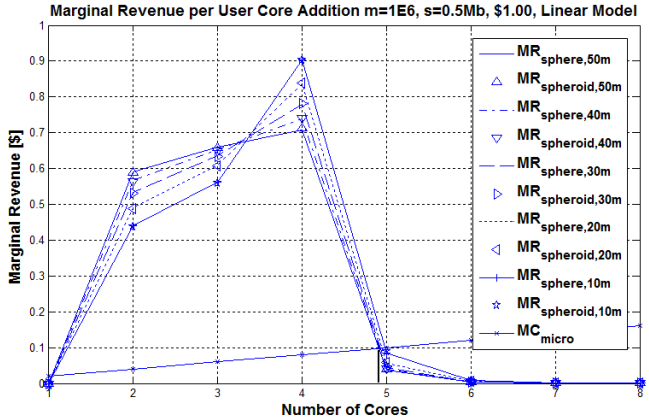

Fig. 10. MR (colored) and MC (black) curves. Linear purchase probability model and transaction fee of $1 per purchase. Optimal number of cores to assign the instance is five.

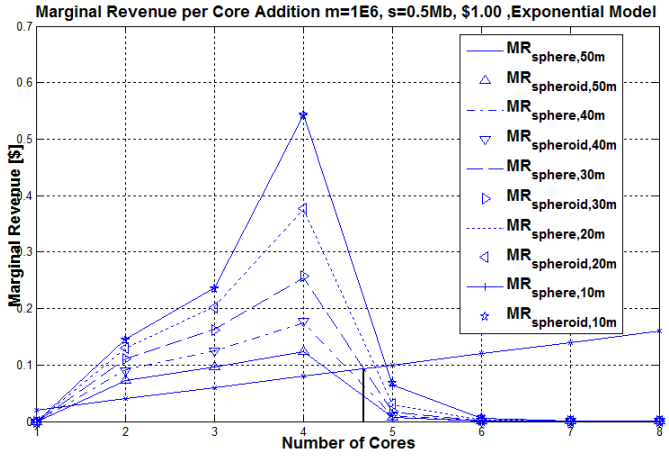

Fig. 11. MR (colored) and MC (black) curves. Exponential purchase probability model and transaction fee of $1 per purchase. Optimal number of cores to assign the instance is five.

Acknowledgment

This work was supported by NSF I-Corps Grant #1313570, entitled "*Proximity - Let the cloud help you!*"

References

[1] Donald Kossmann, Tim Kraska, and Simon Loesing, "An evaluation of alternative architectures for transaction processing in the cloud," in *SIGMOD '10 Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, Indianapolis, 2010, pp. 579-590.

[2] Eucalyptus AWS-Compatible Private Cloud. [Online]. www.eucalyptus.com

[3] R. W. Sinnott, "Virtues of the Haversine," *Sky and Telescope*, vol. 68, no. 2, p. 159, 1984.

[4] T. Vincenty, "Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of nested equations," *Survey Review*, vol. XXIII, no. 176, p. 88–93, April 1975.

[5] Nick Carey, "Establishing Pedestrian Walking Speeds," Portland State University, Portland, 2005.

[6] Amazon Web Services EC2 Instance Types. [Online]. http://aws.amazon.com/ec2/instance-types/

[7] Amazon Web Services Simple Monthly Calculator. [Online]. http://calculator.s3.amazonaws.com/calc5.html