Enterprise Applications in the Cloud:
A Roadmap to Workload Characterization and Prediction

Leonid Grinshpan, Oracle Corporation (www.oracle.com)

Subject

Enterprise application (EA) capacity planning methodology based on queuing models provides reliable estimates of the cloud resources needed to satisfy dynamically changing business workloads [*Leonid Grinshpan. Solving Enterprise Applications Performance Puzzles: Queuing Models to the Rescue, Willey-IEEE Press, 2012, http://tinyurl.com/7hbalv5*]. The biggest challenge in the methodology implementation is a quantitative characterization of EA transactional workload that represents input data for EA models. This article provides a road map to workload characterization and its prediction by:

- Identifying the constituents of EA transactional workload and specifying the metrics to quantify it.
- Reviewing the technologies generating raw transactional data.
- Examining Big Data Analytics ability to extract workload characterization from raw transactional data.
- Assessing the methods that discover the workload variability patterns.

Cloud computing is becoming the most attractive technology for deployment of enterprise applications (EA) because of its intrinsic ability of agile adaptation to the variations of the business demands. Performance of cloud-deployed EAs depends on cloud's ability to dynamically redistribute its resources allocated to EA in order to synchronize them with the fluctuations of business workloads. Queuing model based capacity planning helps to find a right balance between demand from the businesses that are using EA, and supply of resources provided by cloud infrastructure.

Transactional workload – a characterization of a business demand for system services

EA business users generate transactional workload that requires allocation of system resources to be processed. The transactions belong to two categories (Figure 1): the ones that are initiated by people (user transactions) and the others that are triggered by all sorts of equipment connected to EA over the networks (operational transactions).

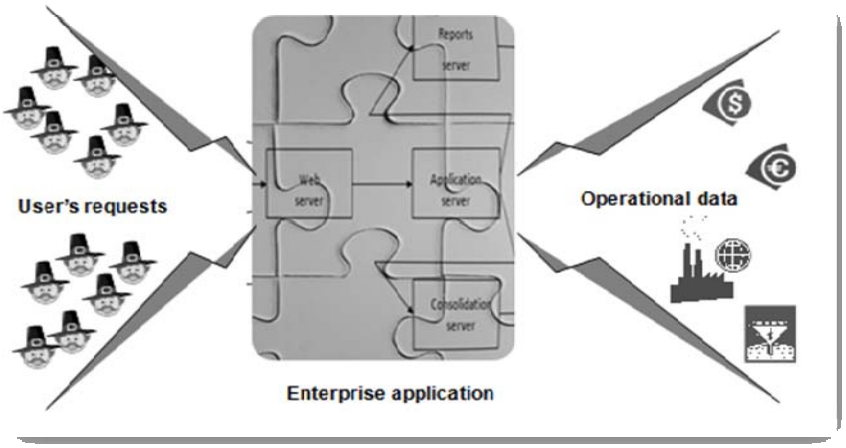

Figure 1 Two sources of transactional workload: user requests and operational equipment.

The intensity of the user transactions at any given time depends on the number of users actively interacting with a system. It also is a derivative of the pace each user submits one transaction after another. The interval between subsequent transactions from the same user can be substantial as a user needs time to assess a reply from an application to a previous request, and to prepare the next one. This time is commonly referred to as think time.

Operational data is generated by devices connected to applications over network (for example, cash registers). Such devices trigger an operational transaction for each event (for example, a sales operation in retail store). Operational transactions let EA keep track of business activities across all company's entities.

Transactional workload is characterized by the following parameters:

- List of transactions (user generated and operational).
- Per each transaction, its intensity expressed in the average number of times a transaction was initiated per one hour by single user or single device.

- Per each transaction, the number of users or devices requesting it.

When being processed by EA, a transaction is consuming system resources. To make such an abstract object as a transaction easier to comprehend, it is quite similar to a car driving on a web of highways and receiving services from highway toll attendants. A highway toll booth is metaphor for a hardware server; an expanse of a toll plaza is a representation of a memory – a car has to occupy some space to be processed by a toll attendant. During rush hours, a car approaching a toll plaza might not find a space to enter it and will wait – the same way a transaction waits for a memory when it is all allocated to other transactions.

Transaction profile – a measure of single transaction demand for system resources

Each transaction initiated by a user or device triggers a multitude of transaction processing activities in different infrastructure components (servers, appliances, and networks). Table 1 describes in orderly fashion all the components of classical three tiered system involved in processing of a transaction that retrieves a sales report.

Table 1

Step	Description of activity	Infrastructure component
1	Transfer a transaction from a user to a load balancing appliance	Network
2	Implementation of a load balancing algorithm in order to direct a transaction to a particular Web server	Load balancing appliance
3	Transfer a transaction from load balancing appliance to a Web server	Network
4	Setting up connection and session between user and system, directing transaction for further processing to Application server	Web server
5	Transfer a transaction from Web server to Application server	Network
6	Analyzing transaction and determining what security and metadata are needed to retrieve a report	Application server
7	Transfer a transaction from Application server to SQL server	Network
8	Retrieving report security data and metadata	SQL server

9	Transferring report security data and metadata to Application server	Network
10	Checking user credentials using report security data and metadata, preparing a request to retrieve data from SQL server needed for report generation	Application server
11	Transferring report to SQL server	Network
12	Fetching data for report generation	SQL server
13	Transferring data for report generation to Application server	Network
14	Generating report	Application server
15	Transferring report to Web server	Network
16	Rendering report	Web server
17	Transferring report to user	Network

As Table 1 indicates, the following infrastructure components are involved in transaction processing: network, load balancing appliance, as well as Web, Application and SQL servers. Each component allocates its resources for transaction processing for a particular time interval. In general, each component has the following assets to be allocated:

Active resources:

- CPU time (data processing)
- I/O time (data transfer)
- Network time (data transfer)

Passive resources:

- Software connections to the servers and services (for example, Web server connections, database connections)
- Software threads
- Storage space
- Memory space

Active resources implement transaction processing and data transfer. Passive resources provide access to active resources. In order to be processed by any active resource, a transaction has to request and get allocated the passive resources. If any of the assets needed for transaction processing is not available because all supply is taken by other transactions, then transaction will wait until an asset is released (indeed, wait time will increase transaction response time).

A consumption of an active resource is measured in the time interval it was serving a transaction. A metric for a passive resource usage depends on passive resource type:

for software connections and threads it is a number of connections / threads; for memory and storage it is a size of allocated memory.

We define a transaction profile as a set of numbers (vector) specifying quantity of each resource consumed by transaction during its processing in hardware components. Table 2 describes profile of transaction *Report*.

Table 2

	Web server		Application server			SQL server			Network	Load balancing appliance
Active and passive resources	**CPU (seconds)**	**Connections**	**CPU (seconds)**	**Threads**	**Memory**	**CPU (seconds)**	**Connections**	**I/O system (seconds)**	**Networking hardware (seconds)**	**CPU (seconds)**
Profile of "Report" transaction	0.01	1	2.5	3	100 KB	0.08	2	0.15	0.09	0.01

Transaction *Report* uses resources of Web, Application, and SQL servers as well as network and load balancing appliance. It spends 2.5 seconds in Application server CPU; before receiving CPU time it requires 3 threads and 100 KB memory to be allocated. The transaction spends 0.08 seconds in SQL server CPU and 0.15 seconds in I/O system only after if it acquired 2 connections to SQL server.

Transactional workload represents a flow of requests to be satisfied by EA. **Transaction profile** is a quantification of a single transaction demand for passive and active resources. **Transactional workload** and **Transaction profile** quantify total demand for EA resources generated by business.

Constituents of transactional data

The menu of business transactions processed by EA is defined during EA development. Each transaction is characterized by a function it is executing as well as by a set of parameters:

Transaction A = {< function>; < parameter 1> ,..., <parameter N>}

Here is an example of a transaction implementing financial consolidation for a particular geographical region, currency, and time period:

Transaction A =
= {financial consolidation; < geographical region>, <currency>, <time period>}

A transaction can be identified by its unique ID. Unique ID enables tracking of a transaction path among system servers and measurement of active and passive resources consumed by transaction during its processing in each server. Information on each executed transaction has to be saved in a log file; in general, this information includes:

- Unique ID
- Name of a user or device that initiated transaction
- Value of each parameter (for example, for **Transaction A** the parameters are: North America, US dollars, year 2012)
- Transaction start date and time
- Transaction total execution time
- For each server a transaction was processed in:
    - Quantity of passive resource 1
    - ……..
    - Quantity of passive resource M
    - CPU processing time
    - I/O data transfer time

- Network data transfer time

Gathering transactional data

Collection of transactional information requires engagement of different technologies. Application development starts from analysis of a business process that identifies all

transactions and their parameters. The analysis is based on the interviews of the key process participants. It produces the flow charts revealing the logistics of a business process, and identifies the steps that can be automated by an application and framed as transactions [http://tinyurl.com/c8yn8fj].

Application instrumentation is the most potent approach capable to deliver transactional data [http://tinyurl.com/7mchth]. A number of instrumentation technologies are adopted by application developers. Among them:

- Application Response Measurement – ARM [http://tinyurl.com/bmxlu83]
- Apache Commons Monitoring [http://tinyurl.com/cm87akg]
- Tracing and Instrumenting Applications written in Visual Basic and Visual C# [http://tinyurl.com/dye2qep]
- Java Management Extension – JMX [http://tinyurl.com/ccyqxtf]

The latest releases of Oracle EAs embrace Execution Context ID (ECID) technology. ECID is a unique identifier that helps to correlate the events associated with the same transaction across several infrastructure components. The ECID value for a particular transaction is generated at the first layer of system (usually a Web server) and is passed down to the subsequent layers. The ECID value is logged (and auditable) in each software component involved in the transaction processing. ECID allows tracking the end-to-end flow of a particular transaction across all EA components (https://blogs.oracle.com/sduloutr/entry/using_execution_context_id_ecid). Here is an example of a message with ECID as it appears in a log file of one of Oracle EA (ECID is highlighted):

[2013-06-06T15:20:10.018-04:00] [FoundationServices0] [ERROR] [01301] [oracle.bi.bifndnepm.workspace.security.RoleChecker] [tid: 609] [userId: <anonymous>] [**ecid: 0000JwQvd83CKuG5Uz1Fic1HYwby0008Zk,0**] [SRC_CLASS: com.oracle.workspace.security.RoleChecker] [APP: WORKSPACE#11.1.2.0] [SRC_METHOD:] Could not resolve role: native://DN=cn=HAVA:0000011ed1cf2e7a-0000-4dc0-0a8f1415,ou=HAVA,ou=Roles,dc=css,dc=hyperion,dc=com?ROLE

Despite the availability of a number of instrumentation methods, it is fair to say that EA instrumentation implementation to the extent that makes EA efficiently manageable, is a rarity. This is because it requires additional coding efforts to be implemented which can cause a serious impact within the always chronically insufficient software development schedule. Fortunately a number of companies are offering the products that are filling the void; they are known as the business transactions management (BTM) systems. As Wikipedia.org explains, BTM systems track each of the hops in the transaction path using a variety of data collection methods, including OS-level sockets,

network packet sniffing, log parsing, agent-based middleware protocol sniffing, and others [http://tinyurl.com/ck3gwby].
Instrumented EAs and BTM systems generate large volumes of raw transactional data that have to be processed to extract workload and transaction profile information representing input data for EA queuing models.

Big Data Analytics – from raw transactional data to workload characterization and transaction profiles

Cloud deployed EAs are designed to process large number of business transactions. For example, saleforce.com estimates that its cloud processes over 1 billion complex transactions every single day (http://tinyurl.com/ntgw9tm). Such large data volumes make Big Data Analytics platforms attractive tools for extraction of workload characterizations and transaction profiles from raw transactional data.

Transactional data is not that different from any other records that the various Big Data implementations are successfully working with. Transactional data mostly is saved as text, XML, or Excel files, as well as in SQL and OLAP databases. A challenge in dealing with transactional data is that the pieces of information belonging to the same transaction are scattered all over hosting platforms and reside in various files on different servers.

Open source Apache Hadoop Data processing framework provides necessary functionality for real time characterization of workload and transaction profiles based on raw transactional records (http://tinyurl.com/3cy6sd). To begin with, transactional data from files and tables spread all around EA have to be loaded into Hadoop. Many ETL tools are available to load transactional data into highly efficient Hadoop Distributed File System (HDFS).

The files in HDFS are broken down by blocks; that allows speedy parallel processing of the same file. A processing is based on MapReduce paradigm: a Map procedure filters data based on unique transaction ID; a Reduce procedure counts a number of transactions with particular unique IDs executed by particular user during one hour time interval. The highly parallelizable nature of MapReduce allows usage of large number of commodity servers generating workload characterization and transaction profiles in real time.

Workload variability patterns

Workload characterization data ~~collected for~~collected for different time intervals represent a basis for discovering the workload patterns and for prediction of its variability.
In order to allocate system resources sufficient to satisfy incoming transactions with acceptable quality, it is necessary to predict fluctuations of hourly workload service demand (WSD). We define hourly WSD as a vector (one-dimensional array):

$$\{hourly\ workload\ service\ demand\} = \sum_{all\ transactions} \{hourly\ transaction\ service\ demand\}$$

where a vector *{ hourly transaction service demand}* is calculated based on vector *{transaction profile}* as shown below for the users (similar calculation takes place for devices)

*{transaction profile} * number of transactions per user per hour * number of users*

It can be seen from the formula that *{hourly workload service demand}* is in direct correlation with the numbers of users (as well as devices delivering operational data) and intensity of their communication with the system. It also depends on the parameters in *{transaction profile}*, particularly on such hardware specification as a speed of CPU, I/O, network and size of available memory and storage.

EA workload variability patterns can be discovered using a number of approaches. A publication http://tinyurl.com/lq5fnka "Workload Analysis and Demand Prediction of Enterprise Data Center Applications" describes analysis of six months workload data for 139 workloads generated by the users of customer relationship management applications. The study concludes that EA workloads typically show a burstiness as well as a periodicity which can be a multiple of hours, days, and weeks. The study presents the methods for deducing workload patterns and assessing their quality.

A paper http://tinyurl.com/lwazjyj "Dynamic Provisioning of Multi-tier Internet Applications" presents a workload predictor that estimates the tail of the arrival rate distribution (i.e., the peak demand) for the next T hours. If T = 1, the methodology predicts the peak demand for the next one hour at the beginning of the hour. The prediction is based on historical workload data collected for each hour of the day over the past few days. A probability distribution is built for historical hourly workload data

and the peak workload for a particular hour is estimated as a high percentile of the workload distribution for that hour.

A web post http://tinyurl.com/lrrm6dg specifies a few of the patterns observed for different EAs in the cloud. Two examples are:

- The workloads that have a relatively short period of activity; they can be generated, for example, by the users of enterprise planning applications that are in use once per quarter.
- The bursty workloads featuring high spikes in demand; they are produced by the users of web retail applications during holiday seasons.

Workload characterization and prediction stack

Described technologies enable workload characterization and its prediction; they form a stack presented in Fig. 2.

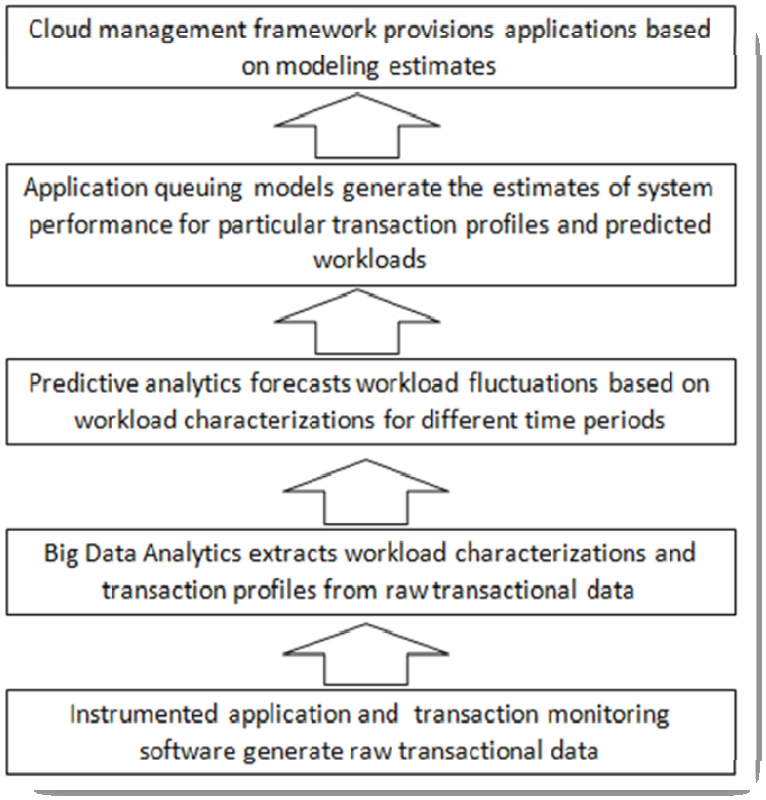

Figure 2 Workload characterization and prediction stack

Transactional workload constitutes input data for an EA queuing model delivering the estimates of EA performance for various infrastructure platforms. Before reassigning EA resources based on modeling estimates, a cloud provider has to evaluate its impact on overall cloud performance. That necessity originates from cloud's nature as a multi-tenant platform. In a dedicated system it is possible to enact a change, then launch EA and evaluate how a change affects EA performance; if change does not deliver needed improvement, we can try a new change and repeat cycle again and again. But reassigning resources for a cloud-based EA in order to improve its performance might degrade performance of other EAs, as well as make cloud to run less profitably. A provider has to use modeling to assess unexpected consequences of a change.

Because EA workload is changing over time, it is necessary periodically to repeat all the steps in the stack presented in Figure 1 to ensure acceptable performance of EA as well as the cloud as a multi-tenant environment.

Takeaway from the article

1. Transactional workload defines a business demand for EA services. It is comprised of the requests coming from EA users as well as from operational equipment.

2. Transactional workload is characterized by:
    - List of transactions.
    - Per each transaction its intensity expressed in an average number of times a transaction was initiated per one hour by single user or single device.
    - Per each transaction a number of users or devices requesting it.

3. Transaction profile is a measure of single transaction demand for system active and passive resources (active resources - CPU, I/O, and network times; passive resources - memory and storage space, number of Web server and database connections, number of software threads).

4. Transactional workload and transaction profile quantify total demand from a business for EA resources.

5. EA instrumentation generates raw transactional data. Big Data Analytics extract transactional workload characterization and transaction profile information from raw transactional data.

6. Workload variability patterns can be identified by analyzing workload characterization for different periods of time; the patterns can be used for predictive capacity planning.

About the author

During his last fifteen years as an Oracle consultant, the author was engaged hands-on in performance tuning and sizing of enterprise applications for various corporations (Dell, Citibank, Verizon, Clorox, Bank of America, AT&T, Best Buy, Aetna, Halliburton, Pfizer, Astra Zeneca, Starbucks, Praxair, Baxter, American Express, etc.).