ITC19/ Proceedings of Poster 2347-2356
Beijing University of Posts and Telecommunications Press

Platform Capacity Assessment and Validation

Hua Xu, Sidney Tang and Yi Chiun Chen

Motorola, Inc.
1501 West Shure Drive, Arlington Heights, Illinois 60004, USA
email: {hua.xu, sidney.tang, y.chen}@motorola.com

Abstract: Accurate assessment of platform capacity is important, not only from product engineering perspective but also from business perspective. Platform capacity analysis serves the purpose of validating product capacity specifications and identifying potential system bottlenecks. It also provides inputs in the development of a system configuration tool or validates the accuracy of an existing configuration tool. Compared to the conventional Erlang-only approach, platform capacity analysis can accurately estimate system configuration by incorporating both traffic behavior and system processing characteristics into the configuration process. In this paper, we present a method of building a flexible model that incorporates traffic profile information in platform capacity assessment. By benchmarking the CPU utilization on various traffic procedures, we built a model that can dynamically estimate platform capacity based on changing traffic profile. We also presented other possible uses of the platform capacity model in areas such as system improvement, system optimization, capacity planning, growth projection, sensitivity analysis, and spare board planning.
Keywords: Cellular, Platform, Performance, Capacity, Configuration

1. INTRODUCTION

Accurate network system configuration is crucial to the business success and operational planning of cellular network operators. Conventionally, system configuration is based on the system capacity (i.e. Erlang capacity) requirement specified by an operator. A configuration tool is typically provided by the system manufacturer and takes the Erlang capacity requirement and/or simple traffic parameters as the inputs and generates a system configuration as the output. System configuration outputs normally include platform-dimensioning specifications such as the number of boards and frames needed in the system, and the link-dimensioning specifications such as the number of E1/T1 links needed.

The Erlang capacity approach may or may not adequately generate configuration specifications that meet the capacity requirement. A system configuration based largely on Erlang capacity requirements and simple traffic parameters may even be misleading at times. It is not surprising to observe that platform capacity is highly dependent on the traffic presented to the platform and the platform's processing capabilities to handle the traffic. And, the platform processing capacity is dominated by the CPU capacity of individual processing boards. In other words, the CPU capacity depends on the "expected cost" of processing in the CPU in handling different procedures associated with a traffic model. [1]

In this paper, we present the results of a study that incorporates traffic models and CPU utilization of traffic procedures for system configuration and dimensioning. We will first discuss the traffic model and the additional information that it provides besides Erlang capacity, which could result in different platform configurations. We will then present and describe a refined capacity model that takes the approach of "expected CPU cost". In section 4, a case study is shown. In section 5, we discuss how the capacity model can be applied in other applications. In section 6, the conclusion of this paper is given.

2. TRAFFIC MODEL

A traffic model contains critical information that determines the system capacity needed in a platform. A traffic model summarizes the expected network traffic pattern during a busy hour (BH). The traffic pattern is described by a set of parameters for some basic system operations. These parameters generally include:

- IMSI attach rate per subscriber
- IMSI detach rate per subscriber
- Location update rate per subscriber
- Call hold time
- Traffic mix, such as
  - Mobile to land call percentage
  - Land to mobile call percentage
  - Mobile to mobile call percentage
- Prepaid call percentage
- Handoff rate
- Call forwarding rate
- Call hold rate
- Call waiting rate
- BHCA per subscriber

Since many of these operations correspond to system computing procedures, in reality, the traffic model is usually represented by the frequency of these procedures being called, or the procedure rates. For instance, the IMSI attach/detach rate is equivalent to frequency that the IMSI attach/detach procedure is called.

Erlang capacity information is incorporated in the traffic model. For instance, the call-hold time of 60 seconds/call and total 1.5 calls per subscriber per busy hour (BH) can be described as:

$$60 \text{ seconds/call} \times 1.5 \text{ calls/subscriber/BH} \div 3600 \text{ seconds/BH} = 0.025 \text{ Erlang/subscriber}$$

Traffic mix information provides much more information than the Erlang capacity. It describes the frequency of individual procedures. For instance, under the same conditions that there are 1.5 calls per subscriber per BH, if 30% of them are MTM, 60% of the calls are pre-paid calls, the procedure rate for MTM of none pre-paid call setup can be calculated as:

1.5 calls/BH/subscriber × 35% × (1-60%) = 0.21 (MTM call setup/ subscriber /BH).

The procedure rate for MTM of prepaid call setup can be calculated as:

1.5 calls/BH/subscriber × 35% × 60% = 0.32 (MTM prepaid call setup/subscriber /BH).

If the MTM percentage is decreased from 30% to 20%, procedure rate for MTM of none prepaid call setup would now be:

1.5 calls/BH/subscriber × 20% × (1-60%) = 0.12 (MTM call setup/subscriber /BH).

The procedure rate for MTM prepaid call setup would be:

1.5 calls/BH/subscriber × 20% × 60% = 0.18 (MTM prepaid call setup/subscriber /BH).

3. REFINED CAPACITY MODEL

It is not counter-intuitive to expect that individual procedures command different CPU resources. The amount of resources needed is dependent on the number of internal and external signaling messages generated and processed in a procedure in a given processor, among other things. However, it is meaningless to take into account just the CPU resource level of a procedure without incorporating the frequency of that procedure. In other words, the CPU resources needed for a given procedure in a particular processor is the product of the CPU utilization and the frequency of the procedure, i.e. the "expected cost" of processing the procedure. The total CPU resource needed in a particular processor is then the sum of all "expected costs" associated with the procedures specified in the traffic model.

With the "expected costs" of a processor understood, the CPU resources for the system in a particular traffic model can then be figured out. This can be accomplished by tallying the expected costs of all the processors in the system.

Since platform capacity, to a large degree, is constrained by the computing capability of individual processing boards in handling a specific traffic profile presented to the system, system configuration that satisfies only the Erlang capacity requirement does not necessarily guarantee an adequate capacity needed in the field. In this section, we present and illustrate a refined platform capacity model that incorporates traffic pattern and CPU utilization information into system configuration and dimensioning in MSC server. With quantitative component processing characterization using a benchmarking technique, we were able to obtain a more accurate capacity estimation based on this model.

3.1. Benchmarking

Benchmarking is a common technique that is used in quantifying the CPU processing capability. It is conducted by sending a specific task repeatedly through the tested CPU and

monitoring the CPU utilization. By benchmarking the CPU cost of each task, the total CPU utilization can be modeled. Benchmark tests can be conducted for all key processor in a laboratory environment. And traffic of a particular profile can be simulated. [2][3]

3.1.1. Laboratory Configuration

To conduct a benchmark study, it is imperative to have a stable laboratory environment. Ideally, the same software and hardware should be employed throughout the testing period. Different software versions may result in different CPU performance results. It requires extra effort to convert benchmark results from one software version to another and it is generally error pruning.

Benchmark test is not a load test and does not need a full system configuration. Two measurements that are tracked during the benchmark tests are the inputs to the CPU and the utilization at the CPU. A small scale, balanced, representative configuration with controlled load to every key processing board is more important than driving the CPU to its limit. When simulating multiple boards performing the same function with equal load sharing, it is important that the loads on all these boards be evenly distributed.

Small test configuration is generally preferred because it can be built with less resources and investment. More importantly, it is easy in a small configuration to generate the necessary testing load that can exercise the testing CPU to a measurable level. The testing load should be carefully designed in each test case. While the load should be high enough so the effect of the load is visible at CPU utilization measurement, it cannot be too high to trigger the overload control mechanism implemented in the tested component.

3.1.2. Benchmark Test

The benchmark test is designed to obtain the CPU utilization per board type per load (procedures/second). Ideally, every major procedure provided by the system should be benchmarked in order to construct the total CPU utilization for a given traffic model. However, in reality, only a subset of the procedures can be benchmarked due to the limitation of lab testing tool and test environment. The more complete the benchmark test set is, the more accurate the capacity model will be.

The benchmark tests are carefully designed so that they can be used to build the capacity model. One procedure at a time is loaded in the system to obtain the CPU cost for that procedure. Typically the load in the benchmark test is designed so that the CPU of the tested board is about 50% utilized.

Take IMSI attach and detach for example. Here is benchmark test procedure:

1. Measure the base CPU utilization of all the processors in the tested component.
2. Start IMSI attach at the designed procedure rate that the configured testing environment can handle.
3. Maintain IMSI attach procedure at the designed rate for a period of time and measure the

To reduce the measurement error, we repeated the same test multiple times and took an average. The CPU utilization measurements were collected for every procedure on every board. This forms the complete set of benchmark test results.

3.2. Board Level Capacity Modeling

With benchmark results, we can build a refined capacity model. The capacity model takes the procedure rate derived from the traffic model as the input and produces a system configuration. Here is the procedure:

- Determine the procedure rates for every procedure from the traffic model.
- Compute the CPU utilization for every procedure of each processing board.
- Compute the total CPU utilization each board by summing up the CPU utilization for all procedures.
- Determine the system configuration from the total CPU utilization.

Let's illustrate this procedure by the following simplified example. Assuming the traffic conditions for the network are: 1.0 IMSI attaches/detaches per busy hour per subscriber, 1.8 location update per busy hour per subscriber, 0.525 MTL call setup/clearing per busy hour per subscribe, 0.525 LTM call setup/clearing per busy hour per subscriber, and 0.45 MTM call setup/clearing per busy hour per subscriber, and assuming the CPU benchmarks for these procedures are 1.5%, 0.3%, 0.5%, 2%, 3%, 4%, 1%, 1%, and 1% respectively.

If the system needs to support 750K BHCA at 25 milliErlang, it needs to handle the IMSI_Attach procedure at a rate of 750K×1.0/3600, which is 208.3 (IMSI_Attaches/second). The CPU utilization for the IMSI_Attach procedure is 208.3 (IMSI_Attaches/second) × 1.5% per (IMSI_Attaches/sec.), which is 312.5%. Similarly, we determine the CPU utilizations for the IMSI_Detach procedure as 62.5%, the LA_Update procedure as 187.5%, the MTL_Call_Setup procedure as 218.8%, the LTM_Call_Setup procedure as 328.1%, the MTM_Call_setup procedure as 375%, the MTL_Call_Clearing procedure as 109.4%, the LTM_Call_Clearing procedure as 109.4%, and the MTM_Call_Clearing as 93.8%.

The total CPU utilization required for this particular call processing board type would be

312.5% + 62.5% + 187.5% + 218.8% + 328.1% + 375% + 109.4% + 109.4% + 93.8% = 1796.9%.

If the target CPU utilization for this kind of call processing board is 80%, and the idle utilization for this board is 5%, the number of call processing boards required to support the 750K traffic model is $\frac{179.9\%}{80\% - 5\%} = 24$.

The number of call processing boards required to support 750K traffic model is 24.

4. CASE STUDY

In this section, we present a case study to illustrate that the refined capacity model can

The capacity model for the call processing board of the MSC server in the CS core is shown in Table 2. Under heavy traffic model, 24 boards are needed to handle the system capacity, while only 19 boards are needed to handle the same system capacity under the light traffic model. A big contributor to this difference is the IMSI attach/detach, which is reduced from 1 to 0.2 per subscriber per BH. As a result, the total CPU utilization is reduced from 375% to 75%, or 300%, which is equivalent to 4 boards. In addition, the combination of the location update rate reduction (close to half) and the MTM call percentage reduction contributes another board reduction.

As demonstrated in this example, the system configuration can vary quite significantly (20% in this example) under different traffic models, even though the Erlang capacity is the same. The Erlang capacity only approach would not account for the traffic behavior difference, thus producing inaccurate result.

5. EXTENSION OF PLATFORM CAPACITY MODELING

The refined capacity modeling not only can provide more accurate system configuration, but also can help in system improvement, system optimization, capacity planning, growth projection, sensitivity analysis, and spare board planning, etc. In the following subsection, we will suggest how the refined capacity model can be applied in sensitivity analysis, spare part planning, capacity expansion, and equipment performance improvement.

In the example in the previous section, the call setup and clearing procedures consists of majority of the CPU usage of the call processing board. For the heavy traffic model, the total CPU of six combined call setup and clearing procedures is 1234.4%, which is 68.7% of the overall CPU usage. The location update only consumes 187.5% CPU, which is about 10% of the total CPU. If there is 10% increase on the call setups and clearing load, almost two more boards need to be added to the system configuration. If 10% increase on the location updates, only 20% more CPU power would be needed and no additional board is needed. We can conclude that the call processing board is more sensitive to the call setup and clearings than to the location updates. Furthermore, among all three kinds of call setups, MTM is the most costly one. If higher MTM call percentage exhibits in the system, more call processing board would be needed. So, for the call processing board, we need to monitor closely the call setup and clearing activities and their mix percentages.

Other type of board, such as the VLR board, the board that handles VLR function, is likely to be more sensitive to the location update than to the call setups. For some system that has excessive location update rate, we need use the refined capcity model for VLR board and configure it properly.

Armed with the CPU usage information for a given traffic profile, operators are in a much better position to configure a more versatile system, define spare part requirements and plan for capacity growth. To build a versatile system, operators can choose to have additional processing boards added to the system if the CPU utilization of the board in the original configuration gets too close to the overload threshold (e.g. 80%). In case of any unexpected

REFERENCES

1. 王爱华, “有关爱立信移动交换设备利用率的探讨”, 邮电规划, 2003 年第一期.
2. John L Hennessy and David A Patterson, Computer Architecture A Quantitative Approach, Morgan Kaufmann Publishers, Inc., 1990.
3. Raj Jain, The Art of Computer Systems Performance Analysis, John Wiley & Sons, Inc., 1991.