DOI: 10.1049/smc2.12026

GUEST EDITORIAL

Scalable computing systems for future smart cities

1 | INTRODUCTION

As I see it, there are three primary computing systems requirements for scalable cities:

- Ease of use.
- Economical and scalable computing infrastructure, including the ability to scale a computing system up or down, as needed, preferably dynamically and automatically.
- High reliability.

I will discuss each in turn, but first, a bias. Scalable cities are first and foremost about people, not about computers or computing. Of course, these days, computing infrastructure is important, but we should never lose sight of the prime directive. The more time and effort we spend on computing infrastructure, the less we can spend on enriching people's lives.

2 | EASE OF USE

With that out of the way, let me address the issues raised above. First, in terms of ease of use, we could mean the ease with which clients interact with the computer system. That is not what I am referring to. Rather, I am referring to the fact that the scalable city is an enterprise, and like all enterprises, it is most likely running standard third party software packages, and has been doing so for a long time. There is a lot of software inertia present in this model. The last thing I would encourage is to require a lot of software modifications to existing software particularly on a fixed inflexible schedule. Of course, as technologies and new ideas emerge, it is important to be able to integrate these with an existing computing base, but large-scale rewriting of existing software must not be mandated or encouraged. New aspects of the smart cities' technology base must be introduced gradually with a clear cost/benefit analysis. The computer systems chosen to run the smart cities must be capable of running both old and new software without modification. It is also important that the infrastructure used in implementing a smart city not be locked into a single vendor. Using standard servers, standard networks, and standard software is, again, highly desirable.

3 | SCALABILITY

Second, let me address the need for scalable computing. Needs change as smart cities evolve. It would be very desirable to preserve investments in computing infrastructure by allowing that infrastructure to support more computing over time without having to invest in the latest shiny new hardware offering. Further, investments that allow an existing hardware technology base to grow and evolve, without having to rewrite software are highly desirable. It would be even better if the system itself can automatically expand and contract due to the demand placed on it, month to month, week to week, day to day, or even at finer levels of granularity. This is well within the state of the art.

You might think I am talking about 'the cloud'. While I do not rule it out, using the cloud has a high potential for locking in customers, as discussed earlier. This is not only true for the hardware that is used, but also the reliance on a set of software packages that only run in single branded cloud vendor's environment can be disadvantageous, since ultimately, the cost of switching away from one vendor to another can be very high or even practically impossible. The marginal costs of using a single cloud vendor can be very high over time due to the vendor's increasing infrastructure costs that often are directly passed along to satisfy shareholder expectations.

4 | RELIABILITY

The third point I wanted to make has to do with reliability. If the smart city is going to rely on its smart city infrastructure, it must be highly reliable and highly available. You might think this comes for free. After all, aren't servers getting more and more reliable? In short, it is becoming increasingly apparent that the answer to this is 'no'. In fact, it is the opposite. As we include more memory in these servers, and increase the density of semiconductors, reliability is decreasing. Part of this has to do with process geometries, part of this has to do with higher utilisation, and part of it has to do with the ability of semiconductors to monitor their own behaviour but not take corrective action when disruptive events are anticipated. When heavily loaded hardware servers fail, the 'blast radius' can become very problematic. Restarting a server can be very expensive in terms of downtime, particularly as the amount of memory in a server increases. Further, it may also take time to get the performance of the server back to an acceptable level (e.g., re-warming caches). The ability to dynamically scale also has an impact on reliability and security.

From a security standpoint, we know that it is important to apply security upgrades on a regular basis, but if it means taking down multiple running systems to upgrade components, this will often not happen according to the desirable fixed schedule. Fortunately, there are solutions to this problem as well. We now can detect a very high percentage of anticipated potential hardware errors, like correctable error correcting codes (ECC) errors predicting non-correctable ECC errors, increasing error rates in network interface cards (NICs), rising temperatures indicating fan failures, and deal with them without having to take mission-critical systems offline.

The dynamic scaling ability allows us to not only take a hardware system offline for repair, but also allows us to add additional capacity when a system becomes overloaded (and then revert when it becomes underloaded). These abilities are all possible and desirable. Further, from an economic standpoint to preserve the investment in computing infrastructure it is very advantageous to not require that a system be overprovisioned just to meet some hypothetical peak demand. It is very advantageous to only use as much computing infrastructure as needed, and only when needed. This is not only true for hardware and energy investments, but also for investments in software licences, which are often correlated with hardware capabilities.

5 | HOW DO WE GET THERE?

I personally believe that distributed virtual machines offer the potential to satisfy all the needs I have mentioned. What is a distributed virtual machine? It is a virtual machine that runs on a dedicated cluster of cooperating physical servers interconnected by a standard network, like Ethernet.

To an operating system, it looks exactly like a single physical server, but it is not. Each physical server runs a piece of software called a hyperkernel. When powering up, each hyperkernel instance takes an inventory of all the processors, all the memory, all the networks, and all the storage on each physical server. Then, the hyperkernel instances exchange this inventory information, and use it to create a single virtual machine. One processor boots a standard operating system, which sees all the combined resources of all the physical servers. The operating system does not even know it is running on a cluster. (It is like a dream: how do you know whether you are dreaming or not?) No modifications to the operating system need to be made, and no modifications to any applications need to be made. Further, the virtual resources like guest virtual processors and guest virtual memory can migrate under automatic control by machine learning algorithms and system performance introspection. So, the first goal of simplicity can thus be achieved.

Scalability is achieved by the cooperating hyperkernels implementing the ability to add and subtract physical servers dynamically as needed. This can be explicit, under operator control, or under programmatic control by some oversight software that tracks performance usage information. Thus, the second goal is achieved.

Reliability is achieved in a very innovative way. The various hyperkernels monitor things like dynamic random access memory error rates, temperature fluctuations, NIC error rates and the like. When an impending problem is detected, there is sufficient time to take corrective action. For example, when a problem is detected on physical server *n*, the hyperkernels on all the other physical servers are told not to send any active guest physical pages or guest processors to *n*. An additional physical server may be added to the cluster to maintain previous performance levels. In other words, *n* is *quarantined*. Physical server *n* is directed to *evict* all active guest physical pages and guest virtual processors to other physical servers. When this is complete, physical server *n* can be removed for repair. A similar process can be used for upgrades of hardware or firmware. All this is done without having to modify or restart the operating system, which is unaware that any of this is taking place. Thus, the third goal, reliability, is achieved.

All this can be achieved with competitive performance using technology available today.

CONFLICT OF INTEREST

Authors declare no conflict of interest.

Ike Nassi[1,2]

[1]*TidalScale Inc., Los Gatos, California, USA*
[2]*Computer Science and Engineering, UC Santa Cruz, California, USA*

Correspondence
Ike Nassi, TidalScale Inc., 15466 Los Gatos Blvd, #109-156, Los Gatos, CA 95032, USA.
Email: Ike.Nassi@TidalScale.com

AUTHOR BIOGRAPHY

Dr. Ike Nassi is an Adjunct Professor of Computer Science and Engineering. He is also the Founder, Chairman, and CTO of TidalScale. His career includes success in large companies, start-ups, and academia. Ike was the EVP and Chief Scientist at SAP when the category of in-memory databases was established. He helped start three companies: Encore Computer, a pioneer in symmetric multiprocessors; InfoGear Technology, which developed the first iPhone, and the category of internet appliances and services; and Firetide, an early wireless mesh networking company. Ike served as a Senior VP and Software head at Apple Computer and held executive positions at Cisco and DEC. He a Founding Trustee of the Computer History Museum. Ike holds several awards from the U.S. DOD, and holds multiple patents in several different areas. He is a graduate of Stony Brook University where he holds a BS in Mathematics, and an MS and PhD in Computer Science. Dr. Nassi is an IEEE Fellow, and Senior Life Member of ACM and IEEE.