‘Cloud’ computing in simulation

Dr Carlo Kopp

THE Moore’s Law driven commodification of computing hardware over the past two decades has produced explosive growth in the area of distributed computing, encompassing the technology of clusters, grids and clouds. This technology permits aggregation of hundreds, thousands or more processing cores to perform large computational tasks, and while initially used for scientific and engineering simulations it is increasingly expanding into other areas of computing.

Clusters, grids and more recently, clouds, have demonstrated great utility and have considerable potential for handling large computational loads previously out of reach for single processor machines, including traditional supercomputers. This in turn has led to numerous predictions of a revolution in computing, and public claims that this technology will reduce the demand for skilled computer scientists, especially those with technical programming skills in operating systems, networks, and systems integration.

The material reality of programming a distributed computer system with numerous processing cores, usually geographically scattered across multiple sites and linked through high speed network connections, is however quite different to the optimistic predictions of the non-technical advocates of this technology. In general, making software run reliably, robustly and quickly across a cluster, grid or cloud actually requires personnel with special skills.

If applications are successfully developed or rehosted to a cluster, grid or cloud computing system the resulting performance gains can often be remarkable. If the application is readily parallelised and thus compatible with a distributed computing system, performance gains directly proportional to the number of computing cores can actually be achieved.

To date this technology has produced its greatest impact in large scale scientific simulation work, spanning areas such as computational electromagnetics, computational aerodynamics, molecular biology, chemistry, pharmacology, geophysics, nuclear physics, astronomy and astrophysics. In recent years it has also penetrated into very large scale web based search and retail engines, with major players like Amazon.com, Google.com and Microsoft offering a cloud computing service product, based on their in-house distributed computing environments.

As with any emerging or maturing technology with great potential, distributed computing, especially cloud computing, has stimulated a cacophony of commercial hype around the technology, and unprecedented numbers of trademark applications centred on names incorporating ‘cloud’ or ‘clouding computing’. This should come as no surprise, as similar behaviour was behind the high technology stocks bubble, which burst around the turn of the millenium producing an unwarranted slump in the sector as wary investors steered away from any and all technology products. Given the failures in due diligence and the propensity to believe hype that preceded this event, there is every reason to expect another cycle of irrational investment behaviour as distributed computing technology matures and penetrates the market. The technology may be sound, but the marketing and sales around it is usually not.

The idea of providing computing power as a ‘utility’ product in the manner of household or industrial electricity, gas, or water is not new but until recently the technology was not available to provide computing power as a ‘utility’ product on demand.

The first community to pursue the use of distributed computing systems were research scientists and university academics working in areas where scientific simulations demanded vast computing power, but budgets to buy supercomputers or time on other peoples’ supercomputers were simply not available. The result of these pressures was the creation of the first computing ‘clusters’ nearly two decades ago. A typical early cluster was constructed by stacking dozens of commodity personal computers, with monitors and keyboards removed, on metal racks and connecting them together using high speed commodity network cabling. Software was then used to make use of the equipment for shared tasks. Broadly these fell into the categories of ‘parametric computing problems’, where the same program needed to be run tens, hundreds or thousands of times over, with slightly different inputs or ‘finite element’ problems. Using this technique a single large calculation was executed by dividing it into many small computational tasks, each of which communicated the results of its computations with a neighbouring task.

Clusters produced spectacular results on well suited computational problems but unspectacular results on not so well suited problems. This is due to an effect first described by supercomputer architect Gene Amdahl during the 1960s, and now known as ‘Amdahl’s Law’.

Amdahl observed that all computations in any computer program can be divided into two categories. One category he labelled as ‘parallel’ were computations that were independent of any other computations in the computer system and thus did not have to wait for other computations to complete. The other category labelled ‘serial’ were computations that depended upon the results of other computations, and thus had to wait their turn until predecessor computations were finished. The end result is that many computational problems are difficult or even impossible to speed up by dividing the work across many computing cores.

The chart showing Amdahl’s Law is based on real data collected on a live database server system around a decade ago. Less than two per cent of the workload that qualified as ‘serial’ effectively slowed this system.

Amdahl’s Law is universally true regardless of the size of the computer system. If your computational problem agrees with Amdahl’s Law you will benefit from using clusters, grids or clouds, if it does not then you need to look at different ways to compute the problem.

A key factor for grids and clouds is that the effect of any ‘serial’ components in a computing workload is severely magnified by any delays in network communications between computing cores in the system. The slower the network connections, or the greater the geographical distances, the worse the problem becomes.

So, as always with real world systems, there are no ‘free lunches’ to be had – if the computational problem falls foul of Amdahl’s Law and does not fit the distributed computing model, no amount of wishful thinking or hoping will make it otherwise.

Military Applications of Distributed Computing

Distributed computing systems are now well established tools used in research and development, and in some areas also as computer aided design tools. Where this research or development is in a military application, the technology can be said to have a direct military use. Areas such as computational electromagnetics, used to model the behaviour of antennas, or stealth shaping, is one beneficiary. Computational fluid dynamic modeling – whether involving the behaviour of aircraft, jet engines, submarine propellers or warship hulls – is another. No differently, the technology is applicable to finite element modeling of structures, whether these are reinforced concrete bunkers, aircraft, ground vehicles or warships.

What Moore's Law and maturation of the software systems and tools for distributed computing will do over time is to make advanced software simulation techniques far more accessible. Smaller design or research teams, smaller companies, and indeed smaller nations will be able to tackle problems that historically demanded more personnel and much greater resources. The upside of this is that research and development projects previously infeasible will become feasible. The downside is that this technology is based on globally accessible commercial products, so rogue states, non-state actors, or 'peer-competitor' states like China and Russia will benefit equally to Western nations.

Distributed computing technology presents other opportunities in military applications, yet to be exploited, and combat simulations are one example. The idea of using grids or clouds for multi-player graphics rich gaming has been explored at length, and is typically taught in university courses as a case study application of distributed computing. Replace the multi-player game with multi-player combat simulation, and you can conduct large scale military exercises in virtual space. While this is not the same as deploying an armoured division to a shooting range, or a fighter wing to a Green Flag exercise, it does present real training value in a manner that is impossible to otherwise affordably perform.

Other potential applications exist in the Intelligence, Surveillance and Reconnaissance domain. This technology is viable for large intelligence collection databases, and has much potential in areas such as datamining.

Operations Research modelling and analysis has historically been computationally intensive, moreso if large repeat simulations are performed where different battles are replayed with different initial conditions or parameters. This is one area of simulation which is almost ideally suited to the distributed platform, whether it is a cluster, grid or cloud.

Applied judiciously, distributed computing technology can provide a valuable advantage to the military user.

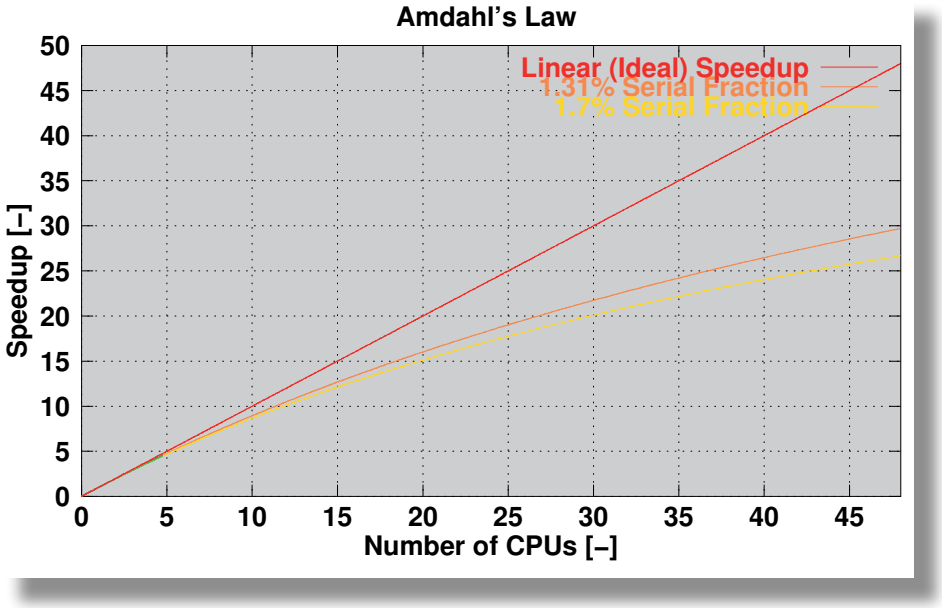
Technology of Clusters, Grids and Clouds

The technology employed to construct distributed computing environments has evolved considerably, and in the process grown in complexity considerably.

A simple cluster could be constructed using commodity off the shelf computing hardware and operating systems, the latter being either Open Source products such as Linux or *BSD, or commercial operating systems offerings including NT, Solaris, AIX, HP/UX or OSF.1.

This provided the basic computing platform to run user applications, compiled and linked to operate on the platform in question. This basic platform did not however provide the capability to manage and run computing jobs doing the actual work.

Products used for this work fell into two categories, either 'parametric computing' tools such as the Australian developed Nimrod and its commercial offspring Enfuzion, or software libraries for managing messaging between individual user tasks, an example being the now defacto standard Message Passing Interface or MPI. While the former often requires no modifications to user application software, the latter must be fully integrated into the user software by a programmer who understands MPI.



Grids are far more complex than clusters, as the operating environment must be capable of managing user jobs across hundreds or thousands of cores, often located at geographically different sites. The intent in a typical grid operating environment is that users can submit tasks from arbitrary sites with access to the grid, these tasks are then distributed across the grid to available cores, with robust security and mechanisms for balancing workloads across the grid.

The key to grids is the availability of 'grid middleware', which is software installed on every machine forming part of the grid. The middleware manages the distribution of tasks across the grid, as well as handling user authentication, security and all other necessary housekeeping chores.

The mostly widely used grid environment is an Open Source product named the Globus Toolkit. A party intending to create a grid simply starts by installing the Globus middleware on all machines to participate in the grid, and then configures the middleware in the intended fashion.

The Globus toolkit is not the only way to play the grid game. Some proprietary software packages provide their own proprietary 'grid-like' mechanisms, often incompatible with Globus, but more than often replicating much of its functionality. The Wolfram Mathematica mathematical modelling software widely used in the hard sciences community has an optional and proprietary grid mechanism, which allows a single desktop or deskside machine to set up a grid across some, licence locked, number of other machines to run Mathematica computations.

Importantly, grids and clusters can typically co-exist. A common practice is to set up a local cluster for local computing tasks, but then install grid middleware such as Globus to permit any or all of the machines forming the cluster to participate in a larger grid. If the cluster is lightly loaded with user tasks, uncommitted cores can be made available for use across such a grid. Many university clusters are configured in this fashion.

Clouds are technologically in many respects the most complex of the distributed computing schemes.

A typical Cloud exploits emulation software. Such emulators run as a program under a commonly used operating system such as Linux or one of the Windows variants, and emulate a specific item of hardware, such as an Intel based personal computer with a specific hardware configuration. A single high performance machine running for instance Linux can then emulate several lower performance personal computers, each of which itself might be running a copy of Linux, Windows, or indeed any arbitrary operating system compatible with that hardware.

In a cloud environment, a user might rent one hundred 'virtual IBM PCs' of a specific hardware configuration, to run his or her workload, using the operating system of choice and application of choice. While these might be current operating systems and applications, they might also be obsolete operating systems and applications that are no longer compatible with contemporary off the shelf computer hardware. In this sense clouds running emulated virtual machines do provide an escape route from the obsolescence of software and hardware – dinosaur applications on dinosaur operating systems can be executed on virtual machines, which are themselves software emulations of dinosaur hardware. Where the cost of rewriting, porting or otherwise resuscitating no longer supported software applications is prohibitive, or impossible as the provider of the code no longer exists and the original source code is lost, emulation is a valuable tool.

Installing copies of emulation tools such as VMWare or Parallels on hundreds of machines is not enough to make up a cloud. A software environment must be provided which can manage user tasks, which includes launching the emulator software, and loading and booting the user's operating system and runtime applications on the emulated virtual machine. At present there are nearly twenty software products in the market being offered as "cloud platforms".

The notion that distributed computing will do away with the need for highly skilled and well educated computing professionals makes little sense once we explore what it actually takes to set up, configure and manage such systems. The opposite is true, and twelve months ago this author participated in a major rewrite of a whole university computing degree curriculum to simply ensure that graduates would be equipped with the basic skills to survive in a future world of distributed computing systems.