This work is partially funded by the European Commission under FP7-ICT-2009.8.1, GA no. 248465

S(o)OS Newsletter

Issue N°1 – January 2011

Dear *reader*,

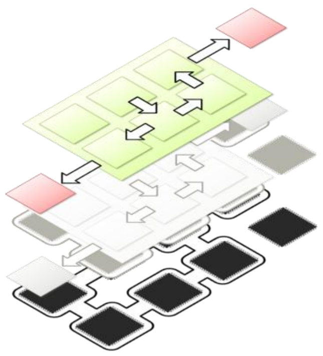
welcome to the first issue of the S(o)OS Newsletter. The Service-oriented Operating Systems (S(o)OS) European Project addresses the needs of future massively-parallel and distributed systems by drawing from the principles and lessons learned in the domains of service-oriented architectures (SOA), Grid and distributed computing. This first newsletter has the purpose of spreading the word about the research issues that are being investigated in the project and the approach we are following for addressing them.

INTRODUCTION

Processor and network architectures are making rapid progress with more and more cores being integrated into single processors and more and more machines getting connected with increasing bandwidth. Processors become heterogeneous and reconfigurable, thus allowing for dynamic adaptation to specialised needs. In the future, thousands of billions of devices may be connected to form a single computing system.

Current programming models are unable to cope with these developments, as they are too tightly coupled with the underlying device structure. Furthermore, complex, non-aligned middleware and operating systems make the programming model unnecessarily inefficient. In order to realise efficient programmability of large scale (terascale) devices by experts and average developers equally, a complete new approach to handling these types of devices across all layers is required.

The S(o)OS project is addressing future distributed systems on the level of a holistic operating system architecture by drawing from Service Oriented Architectures and the strength of Grids. This decouples the OS from the underlying resource infrastructure, thus making execution across an almost unlimited number of varying devices possible, independent from the actual hardware.

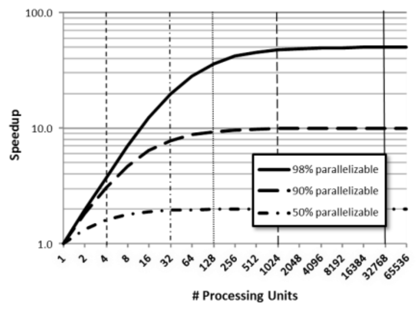

mapping code to infrastructure

S(o)OS will allow for automatic distribution of code parts across such a resource fabric by investigating means to execute processes, threads and parallel applications across the resources in a way that addresses both code requirements and resource availability, thus improving overall performance.

S(o)OS intends to enable even average developers to cope with large, widely distributed infrastructures. The project therefore examines means for run-time code analysis, segmentation and distribution across the infrastructure. This will range from automated analysis of the code parallelism to powerful extensions for experienced developers that will allow to specify, for example, communication and dependency requirements across threads or segments.

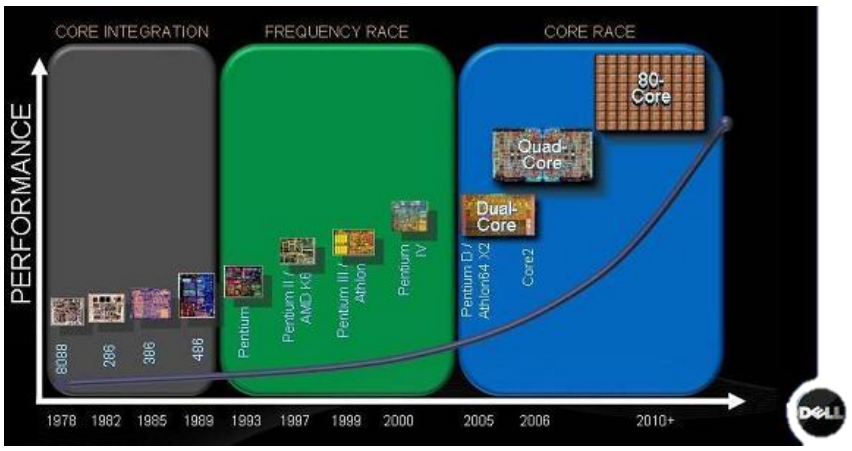
THE PROBLEM

The challenging goal aimed to by the S(o)OS project is the one of designing a software stack that is suitable for future and emerging massively parallel, distributed and heterogeneous systems. The attention is strongly focused on the *architecture of the Operating System* and its kernel(s), and the interactions with the applications through suitable *novel programming paradigms*.



Amdahl's law of scalability

In the nowadays computing era, we have still a clear distinction between the *general-purpose computing* (GPC) and the *high-performance computing* (HPC) worlds. In GPC, programmers use traditional development techniques based mainly on sequential programming, and use traditional Operating System services managing to achieve a quite good usage of the underlying computing power, as available on traditional hardware with one or a few processors/cores. In HPC, particularly skilled developers use dedicated middleware, OS services, special parallel programming paradigms and hardware-specific optimisation techniques, in order to achieve very high saturation levels of the underlying massively parallel and vectorial machines. Such exploitation levels would be nearly impossible to achieve on these platforms recurring to the traditional and widely used development techniques of the GPC world.

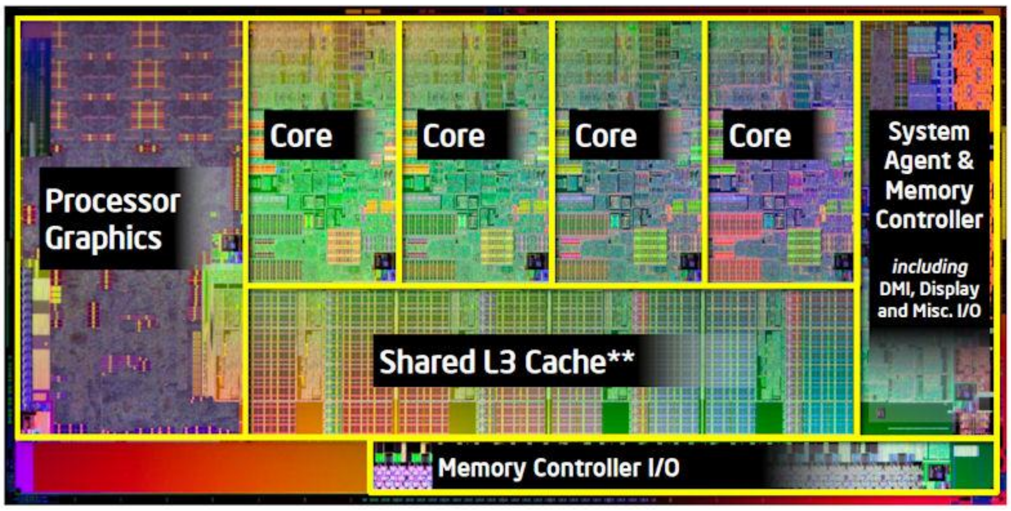


[source: Dell Computers]

development of parallel architectures

However, the future and emerging trend in hardware development is going towards massively parallel architectures which are going to be more and more used for general-purpose computing as well. For example, in the Cloud Computing domain, there is an increasing interest by Infrastructure-as-a-Service providers in using big machines with many cores for the provisioning of virtualised infrastructures to customers. We need also to mention the promising trend in multi-core computing for embedded systems due to the foreseen savings in power consumption. Therefore, the worlds of GPC and HPC are going to get closer and closer into the future, thus we need software development techniques that allow the majority of the programmers to take advantage of the computing power available on the new generation hardware. This involves new redesigns at all levels of the software stack, starting from the Operating System itself. This has to play a better job in mediating the interac-

tion between applications and the hardware by providing a set of abstractions which need to account for the novel massively parallel availability of computing power and resources. Furthermore, the OS and particularly its kernel needs to get rid of *scalability problems* that arise when a traditional OS tries to manage too many resources.
Indeed, one widely recognized issue is that the Operating Systems mainly used for general-purpose computing nowadays have serious bottlenecks when trying to scale to platforms with a high number of cores. This is already evident with servers readily available today with up to 16, 32 and 48 cores. New kernels for the future machines need to be highly modular, heterogeneous, and they need to provide non-centralised, heavily distributed services to applications. To this purpose, their own internal architecture cannot exhibit centralised elements at risk of becoming serious bottlenecks when trying to scale to thousands of cores, but it needs to be heavily distributed as well.



[source: HEXUS]

Intel's Sandy Bridge already integrates vector cores with general purpose cores

One more problem is the heterogeneity of the hardware platforms. These cannot be easily exploited by the upper layer software stack, unless a big effort is put in place for rewriting major parts of the OS, middleware and applications.

THE SOLUTION: S(O)OS

In order to cope with the complex and challenging problem of designing a suitable software stack for easing the task of programming complex massively parallel systems of tomorrow, the approach followed in the S(o)OS project comprises a set of elements which can be summarised as follows:

● **Hardware description and simulation**: in S(o)OS, proper hardware description languages and simulation tools are being developed in order to build a concrete framework over which to experiment the effectiveness of the solutions that will be proposed.

● **Communication models**: one of the key factors that are being investigated in S(o)OS is the type of communication models and protocols that may effectively be used for the data distribution and communications among the unprecedented number of computing elements foreseen inside a single chip and composing a distributed system.

● **Distribution of the execution**: a particularly crucial role is played by the work and data distribution logic and the scheduler(s) that act inside the Operating System, so as to achieve maximum performance and/or meet application requirements. This becomes even more critical when the deployed applications need to comply with precise and possibly strict timing, latency and Quality of Service requirements.

- **Programming paradigms**: the type of programming paradigm that may be used by developers on future parallel platforms is crucial to the degree of concurrency, saturation of the computing power achievable on such platforms and the scalability of software. Moreover, it novel programming paradigms are needed to ease the task of developing efficient and scalable applications on heterogeneous future platforms.

The investigations in the above elements will allow for the design of an architecture of Operating System and its kernel(s) that will be suitable for playing the needed mediation role in future massively parallel architectures. The OS will expose interfaces with the essential level of abstraction from the specifics of the hardware which will allow application developers to achieve more easily a good exploitation of the underlying computing power and concurrency level.

MORE INFORMATION

More information on the S(o)OS project can be found on the official website:
http://www.soos-project.eu/.
S(o)OS belongs to the TERACOMP initiative which includes other projects investigating on future and emerging terascale computing platforms.

http://cordis.europa.eu/fp7/ict/programme/fet_en.html