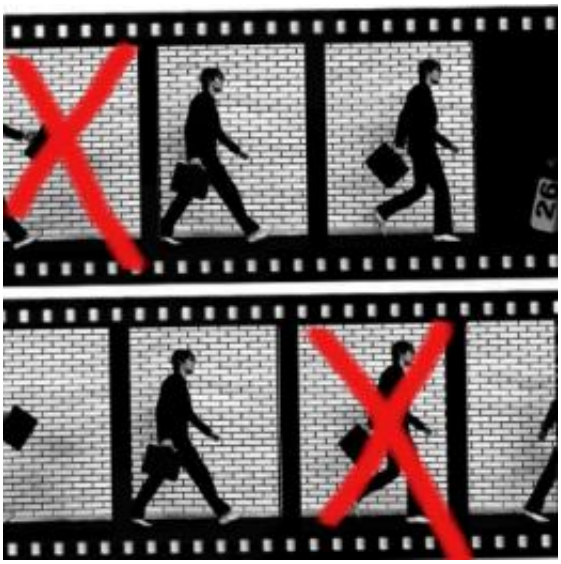
The next operating system

February 24 2011, By Larry Hardesty



Graphic: Christine Daniloff

At the most basic level, a computer is something that receives zeroes and ones from either memory or an input device — like a keyboard — combines them in some systematic way, and ships the results off to either memory or some output device — like a screen or speaker. An operating system, whether Windows, the Apple OS, Linux or any other, is software that mediates between applications, like word processors and Web browsers, and those rudimentary bit operations. Like everything else, operating systems will have to be reimagined for a world in which computer chips have hundreds or thousands of cores.

Project Angstrom, an ambitious initiative to create tomorrow's

computing systems from the ground up, funded by the U.S. Defense Department and drawing on the work of 19 MIT researchers, is concerned with multicore computing at all levels, from chip architecture up to the design of programming languages. But at its heart is the development of a new operating system.

A computer with hundreds of cores tackling different aspects of a problem and exchanging data offers much more opportunity than an ordinary computer does for something to go badly wrong. At the same time, it has more resources to throw at any problems that do arise. So, says Anant Agarwal, who leads the Angstrom project, a multicore operating system needs both to be more self-aware — to have better information about the computer's performance as a whole — and to have more control of the operations executed by the hardware.

To some extent, increasing self-awareness requires hardware: Each core in the Angstrom chip, for instance, will have its own thermometer, so that the operating system can tell if the chip is overheating. But crucial to the Angstrom operating system — dubbed FOS, for factored operating system — is a software-based performance measure, which Agarwal calls "heartbeats." Programmers writing applications to run on FOS will have the option of setting performance goals: A video player, for instance, may specify that the playback rate needs to be an industry standard 30 frames per second. FOS will automatically interpret that requirement and emit a simple signal — a heartbeat — each time a frame displays.

If the heartbeat fell below 30, FOS could adopt some computational short cuts in order to get it back up again. Computer-science professor Martin Rinard's group has been investigating cases where accuracy can be traded for speed and has developed a technique it calls "loop perforation." A loop is an operation that's repeated on successive pieces of data — like, say, pixels in a frame of video — and to perforate a loop

is simply to skip some iterations of the operation. Graduate student Hank Hoffmann has been working with Agarwal to give FOS the ability to perforate loops on the fly.

Saman Amarasinghe, another computer-science professor who (unlike Rinard) is officially part of the Angstrom project, has been working on something similar. In computer science, there are usually multiple algorithms that can solve a given problem, with different performance under different circumstances; programmers select the ones that seem to best fit the anticipated uses of an application. But Amarasinghe has been developing tools that allow programmers to specify several different algorithms for each task a program performs, and the operating system automatically selects the one that works best under any given circumstances. That functionality will be a feature of FOS, Agarwal says.

Dynamic response to changing circumstances has been a feature of Agarwal's own recent work. An operating system is, essentially, a collection of smaller programs that execute rudimentary tasks. One example is the file system, which keeps track of where different chunks of data are stored, so that they can be retrieved when a core or an output device requires them. Agarwal and his research group have figured out how to break up several such rudimentary tasks so that they can run in parallel on multiple cores. If requests from application software begin to proliferate, the operating system can simply call up additional cores to handle the load. Agarwal envisions each subsystem of FOS as a "fleet of services" that can expand and contract as circumstances warrant.

In theory, a computer chip has two main components: a processor and a memory circuit. The processor retrieves data from the memory, performs an operation on it, and returns it to memory. But in practice, chips have for decades featured an additional, smaller memory circuit called a cache, which is closer to the processor, can be accessed much

more rapidly than main memory, and stores frequently used data.

In a multicore chip, each core has its own cache. A core could probably access the caches of neighboring cores more efficiently than it could main memory, but current operating systems offer no way for one core to get at the cache of another. Work by Angstrom member and computer-science professor Frans Kaashoek can help redress that limitation. Kaashoek has demonstrated how the set of primitive operations executed by a computer chip can be expanded to allow cores access to each other's caches. In order to streamline a program's execution, an operating system with Kaashoek's expanded instruction set could, for instance, swap the contents of two caches, so that data moves to the core that needs it without any trips to main memory; or one core could ask another whether it contains the data stored at some specific location in main memory.

Since Angstrom has the luxury of building a chip from the ground up, it's also going to draw on work that Kaashoek has done with assistant professor Nickolai Zeldovich to secure operating systems from outside attack. An operating system must be granted some access to primitive chip-level instructions — like Kaashoek's cache swap command and cache address request. But Kaashoek and Zeldovich have been working to minimize the number of operating-system subroutines that require that privileged access. The fewer routes there are to the chip's most basic controls, the harder they are for attackers to exploit.

Computer-science professor Srini Devadas has done work on electrical data connections that Angstrom is adopting (and which the previous article in this series described), but outside Angstrom, he's working on his own approach to multicore operating systems that in some sense inverts Kaashoek's primitive cache-swap procedure. Instead of moving data to the cores that require it, Devadas' system assigns computations to the cores with the required data in their caches. Sending a core its

assignment actually consumes four times as much bandwidth as swapping the contents of caches does, so it also consumes more energy. But in multicore chips, multiple cores will frequently have cached copies of the same data. If one core modifies its copy, all the other copies have to be updated, too, which eats up both energy and time. By reducing the need for cache updates, Devadas says, a multicore system that uses his approach could outperform one that uses the traditional approach. And the disparity could grow if chips with more and more cores end up caching more copies of the same data.

This story is republished courtesy of MIT News (web.mit.edu/newsoffice/), a popular site that covers news about MIT research, innovation and teaching.

More information: Computer chips' clocks have stopped getting faster. To maintain the regular doubling of computer power that we now take for granted, chip makers have been giving chips more "cores," or processing units. But how to distribute computations across multiple cores is a hard problem, and this five-part series of articles examines the different levels at which MIT researchers are tackling it, from hardware design up to the development of new programming languages.

Part 1. Designing the hardware - www.physorg.com/news217669712.html

Provided by Massachusetts Institute of Technology