John Otto
2007 Winter – Advanced OS
Fabián Bustamante
23 January 2007

R. von Behren, J. Condit, F. Zhou, G. Necula and E. Brewer, Capriccio: Scalable Threads for Internet Services, In Proc. of the 19th ACM Symposium on Operating Systems Principles, Oct. 2003

Summary:
Capriccio is a user-level thread package that uses non-blocking I/O to achieve scalable thread performance, dynamic stack allocation based on a checkpointing system between function calls to efficiently use space allocated to thread stacks, and resource-aware scheduling to dynamically manage at runtime scheduling patterns to maximize utilization of available resources.

Important ideas:
The Capriccio scalability tests were performed against kernel-level thread packages since most user-level thread packages use blocking I/O and would therefore not provide comparable performance. Capriccio minimizes calls into the kernel, improving performance. By analyzing stack space requirements at compile time, and then efficiently allocating memory for thread stack frames at run-time through a checkpointing system, threads need not be instantiated with stack space typically far greater than a thread usually requires. Capriccio monitors blocking calls at runtime to manage resources being consumed and freed, using this information to preferentially schedule threads that are predicted to maximize resource utilization.

Flaws:
It seems as if the predictions and hopes for future work are overly ambitious and optimistic, though this is based on the "attitude" of the comments and not the technical reality of the situation. Regarding the linked stack management, Capriccio's fall-back method to allocating 2MB for a library call with unknown stack requirements simply cannot scale; maybe there is a way to monitor stack use in the library and dynamically adjust the allocated size for future similar library calls. Resource-aware scheduling currently has high overhead because of the inherent stack trace at every blocking I/O call. With the analyses, switching in the middle of the paper from the 2.5 development kernel to the 2.4 release kernel seems awkward, especially since the 2.4 kernel didn't have the same non-blocking I/O call as the 2.5 development kernel; much of Capriccio's performance hinges on the non-blocking I/O implementation, and it feels as if switching kernels invalidates one or both of the sets of results.

Relevance:
Scalable web application servers must handle many requests concurrently. Allowing a programmer to write simple threaded applications and use an advanced user-level thread library to optimize performance facilitates development of scalable, durable application code.