Inter-System Call Parallelism

Jie Liao, *Aaron Carroll, Lin Zhong
{jie.liao, lzhong}@rice.edu, *aaron.carroll@nicta.com.au

Problem Statement

- Current thread model only allows single thread to issue one system call at a time.
- Sequential system call execution becomes a significant performance bottleneck for threads with many system calls.

Solution: Inter-system call (syscall) parallelism

Execute independent syscalls in a SINGLE thread on MULTIPLE cores in parallel.

Example:

a. 0x1c = epoll_wait()
b. read(0x31)
c. sendmsg(0x31)
d. read(0x30)
e. sendmsg(0x30)
f. read(0x27)
g. sendmsg(0x27)
... 
h. read(0x1d)
i. sendmsg(0x1d)
j. 0xe = epoll_wait()

Code snippet from Memcached

Inter-syscall Parallelism Patterns

- Parallel iteration e.g., Memcached and Nginx
- Loop pipelining e.g., tar, cp and dd
- Iterative searching e.g., grep and gc

Inter-syscall Parallelism Is Abundant

- Potential speedup on single-threaded programs:
  - 69%~109% on event-driven servers
  - 22%~42% on some CLI utilities
- Inter-syscall parallelism is not ubiquitous.

Inter-syscall Parallelism Is Achievable

- Client
- Multithreaded
- Quad-core Core-i7
- Server
- 1 worker thread/core
- Quad-core Core-i7

Future Work

- Moving libAsyncOS functionalities to kernel, with techniques such as batching, exception-less syscall invocation, etc.
- Overlapping computation with syscall execution
- Exploring distributed operating system supports for inter-syscall parallelism