Cryptography on multicore systems: an Erlang case

Kenji Rikitake
Academic Center for Computing and Media Studies (ACCMS), Kyoto University
for CSS2010 Session 2A2
20-OCT-2010
Programming on multicore systems

Available number of cores already enormous
Tilera: 512 cores in 2U server < 250W now [1]

Shared-memory coding doesn't work
Minimizing locking is critical and essential
State must be moving around with functions
Message passing will be the (casual) solution

Crypto (RNG, hashing) = full of states

Code with hidden states is not reentrant and unusable in the multicore systems!

My experience of SFMT RNG on Erlang

Making the C code reentrant
  http://github.com/jj1bdx/sfmt-extstate
Writing a pure Erlang version of SFMT
Writing a NIF version of Erlang SFMT
  with the reentrant C code
  ~40 times faster than the pure Erlang code
Available in Github at
  http://github.com/jj1bdx/sfmt-erlang

Acknowledgment
  Kyoto University Super Computer Thin Cluster has been extensively used for the code testing and development of the sfmt-erlang software.
How Erlang BEAM (VM) runs crypto

Based on OpenSSL

More CPU core alloc control by BEAM

Long ago: single-thread linked-in driver
Previous: multi-thread linked-in driver

Now (since R14A) NIF
Native Interface Functions: part of BEAM in C
Mapped onto BEAM processes for max parallelism

Lesson learned: let the application (or the VM), not the OS, control the CPU core scheduling
Crypto for concurrent/parallel systems

Split the en/decrypting data for parallelism

MS Office 2007 SP2 uses 4096-byte cipher block and uses a new key for each [2]

To ensure random access and recoverability

This approach also applicable to multicore crypto

Keeping states and data intact

Keep \{IV, Algorithm, <<Data>>\} together

Algorithm: Key+Crypto / Hash func / RNG, etc.

An example of parallel encryption

Splitting data into multiple blocks enables applying parallelism to encryption

Master IV → IV generator

plaintext data

abcdefghijkl
mnopqrstuvwxyz
yz0123456789
....

parallel encryptors
(e.g., one core for each)

IV generator

IV1 Alg
IV2 Alg
IV3 Alg
IV4 Alg
IV5 Alg
IV6 Alg
IV7 Alg
IV8 Alg

IV1~IV8 set

encrypted data

ehualkdghas
ngtuinvkakjasd
bangigoalriosls
....

Master IV
My questions to the crypto experts

Any parallel-oriented crypto algorithms?
More speed if many cores are fully utilized

Asymmetric multiprocessing needed?

Typical applications: parallel IDS/packet filter
Core assignment for the optimal performance?

Multi-platform compatible algorithms?
not only x86(-64); ARM, Tilera, GPGPU, etc.

The last question: are you ready?