Sun's 3rd generation on-chip UltraSPARC security accelerator

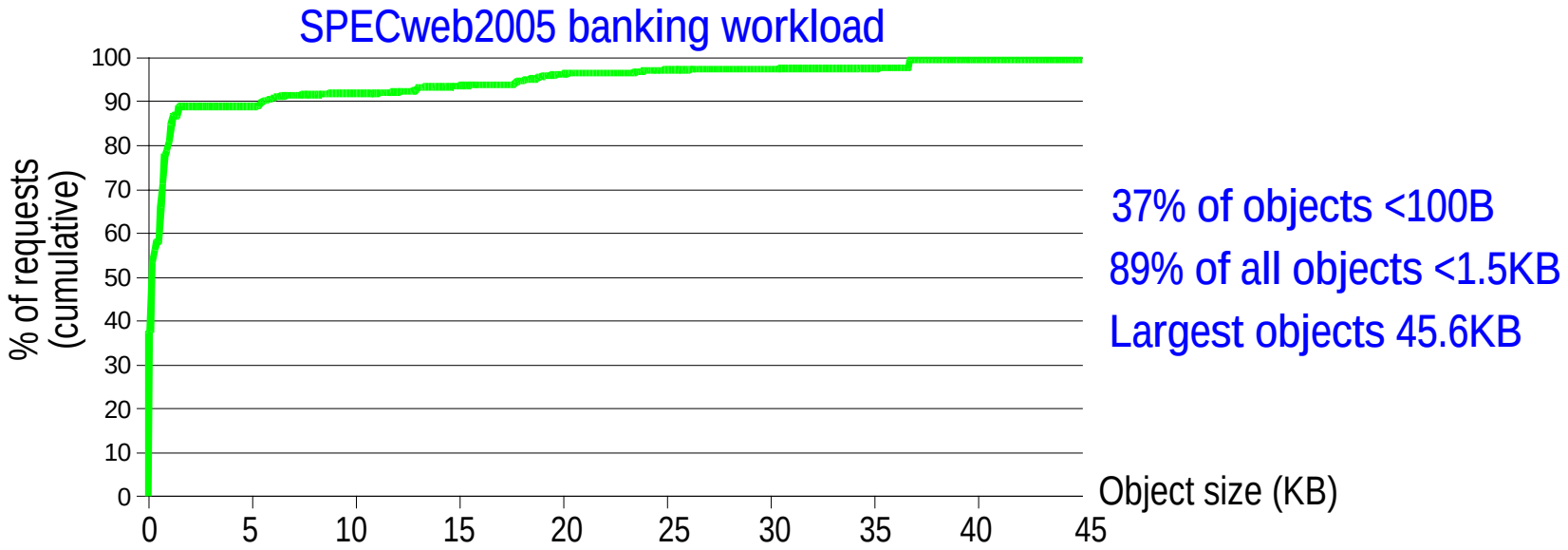
Lawrence Spracklen
Sun Microsystems

Accelerators are evolving

- Security is becoming evermore essential
  - From web servers to databases, from filesystems to networking
- Security is costly from a performance perspective
  - 2X+ slowdowns are commonplace when 'going secure'
  - High cost is hindering adoption
- *Offloading* cryptographic processing to accelerators can virtually eliminate the security overhead
  - Reduces cost of crypto processing by 20X+
  - Zero-cost security?
- Traditional off-chip accelerator approach has significant limitations
  - Benefits can be limited to accelerating RSA operations
- Accelerators have been steadily moving closer to the cores
  - Necessary for effective acceleration in many application spaces
  - Modern processors typically have on-chip support for cryptographic acceleration

Accelerator usage models

- In many applications, the size of most objects processed by accelerators is small



- Phenomenon not just limited to web workloads e.g.
  - IPsec dealing with <1500-byte objects
  - VoIP dealing with ~250-byte objects
- Requires strict control of software overheads associated with using the accelerator

UltraSPARC accelerator evolution

1) UltraSPARC T1 (aka Niagara) processor [2005, 8 accelerators]

- Accelerators target modular arithmetic operations
  - Accelerate public-key cryptography (e.g. RSA algorithm)

2) UltraSPARC T2 processor [2007, 8 accelerators]

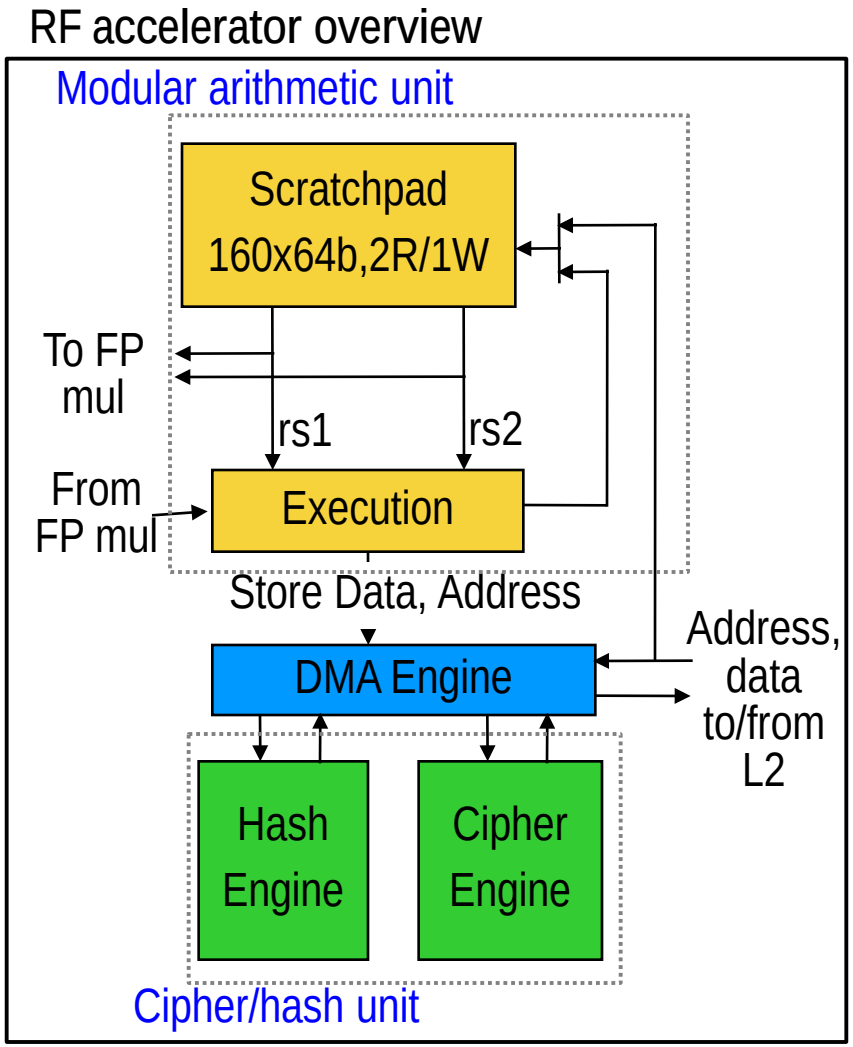
- Accelerators enhanced to also support:
  - Bulk encryption
  - Secure hash
  - Elliptic Curve Cryptography (ECC)

3) Rainbow Falls (RF) processor [16 accelerators]

- Accelerators further enhanced to support:
  - Kasumi bulk cipher
  - SHA-512 (rounding out SHA-2 support) & partial hash support
  - Non-priv 'fast-path' to the accelerators

RF UltraSPARC crypto accelerator

- Accelerators are per core
  - 2 basic sub-units (can operate in parallel)
  - Operate in parallel with threads
- Accelerator is shared by all the core's strands
  - 8 strands per core on UltraSPARC RF
- Accelerators are Hyerprivileged
  - Each strand could be under the control of a different OS
- Accelerators expose a light-weight interface to SW
  - Communication via a memory-based control word queue (CWQ)
  - Requests are fully self-contained
  - Both sync and async operation supported



Rainbow Falls (RF) peak performance

Bulk cipher

Algorithm
DES
3DES
AES-128
AES-192
AES-256
Kasumi

Secure hash

Algorithm
MD5
SHA-1
SHA-256
SHA-512

Public key

Algorithm
RSA-1024
RSA-2048
ECC

- RF provides up to 16 accelerators per processor
- Common ciphers supported (helps SSL, IPsec etc)
- HW peak performance is dependent on object size
  - ~90% of peak for 1KB objects when L2$ sourced
  - ~70% of peak for 1KB objects when DRAM sourced
- Accelerators support common modes of operation for block ciphers (EBC, CBC, CTR, & CFB)
- Hashed Message Authentication Code (HMAC) support
- HW gather support
- HW support for IP checksum and CRC32c acceleration and data movement

Additional RF crypto instructions

- Rainbow Falls (RF) introduces several crypto centric non-priv instructions:

umulxhi

- Returns the upper 64-bits of a 64x64-bit integer multiplication
- Along with new addxc{cc} instructions allows bignum functions to operate directly on 64-bit data chunks

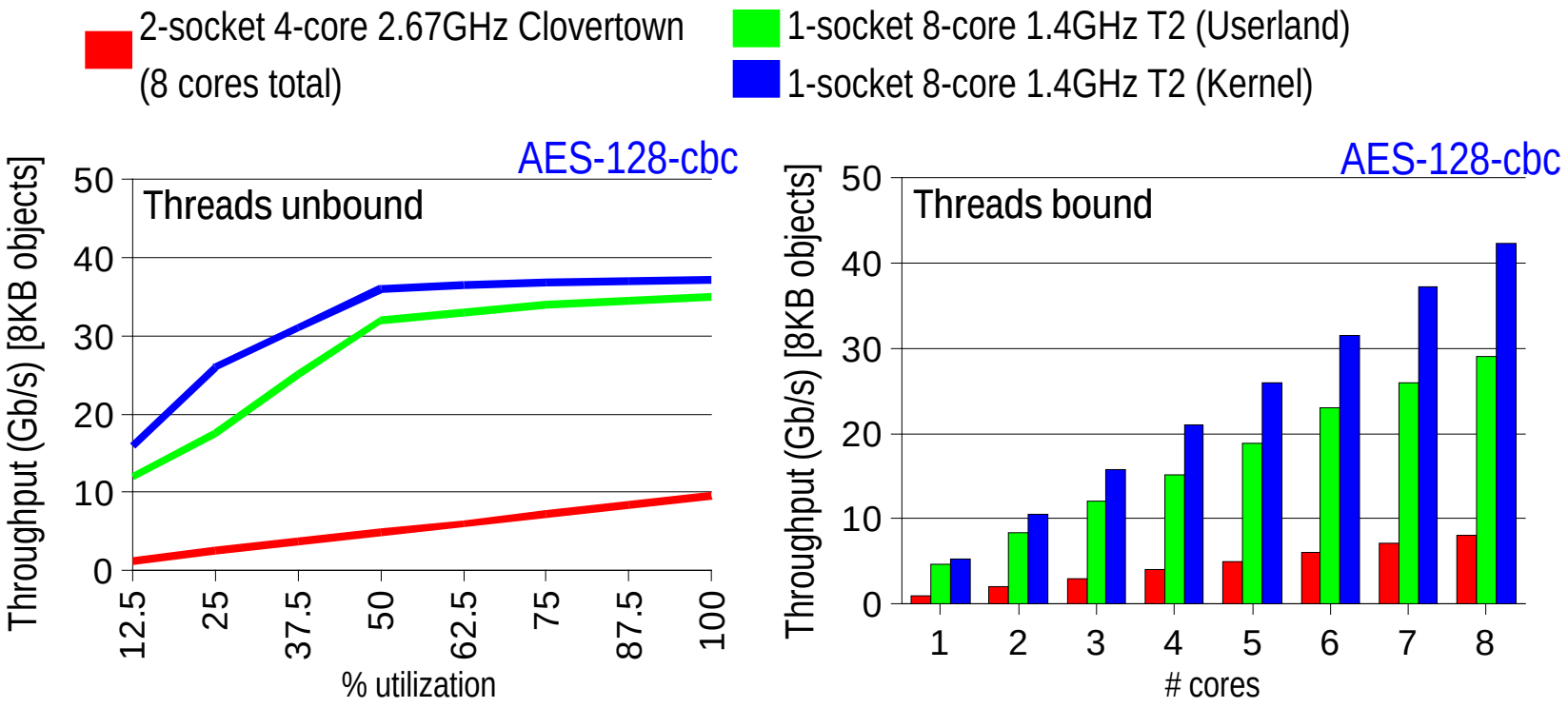
xmulx/xmulxhi

- Can be used to accelerate Galois field computations
- Important for many authenticated encryption algorithms e.g. AES-GCM
- Preferable to use the dedicated accelerators for GF(2^M) ECC operations

- RF multiplier is fully pipelined
- RF also introduces an IP checksum instruction
  - Useful for IPsec acceleration (network card checksum generation not practical)

UltraSPARC T2 accelerator performance (large packets)

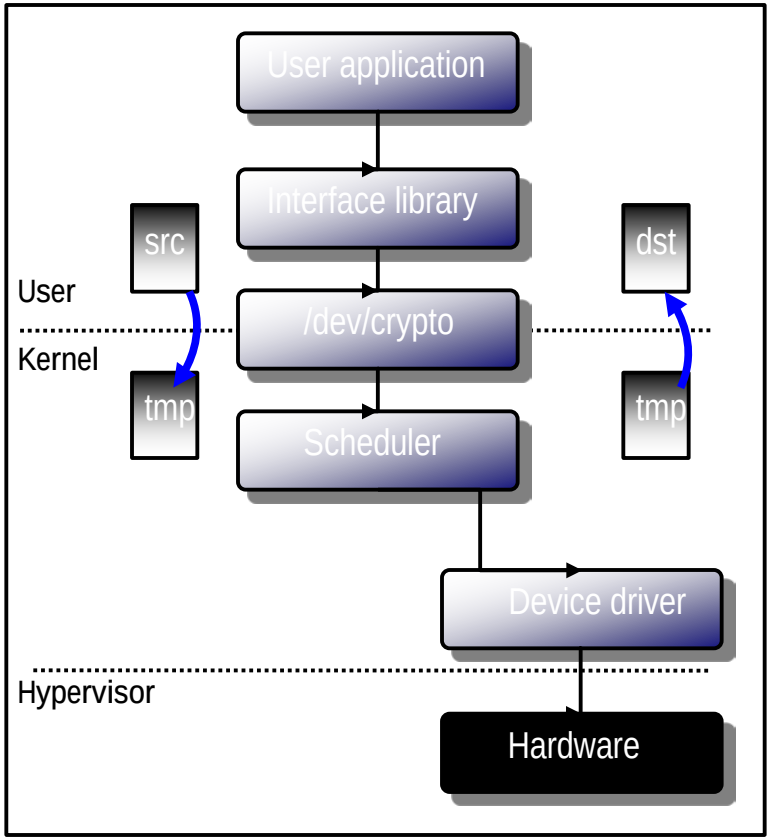
- Accelerators deliver excellent large object performance



- Small packet performance is also critical to customers

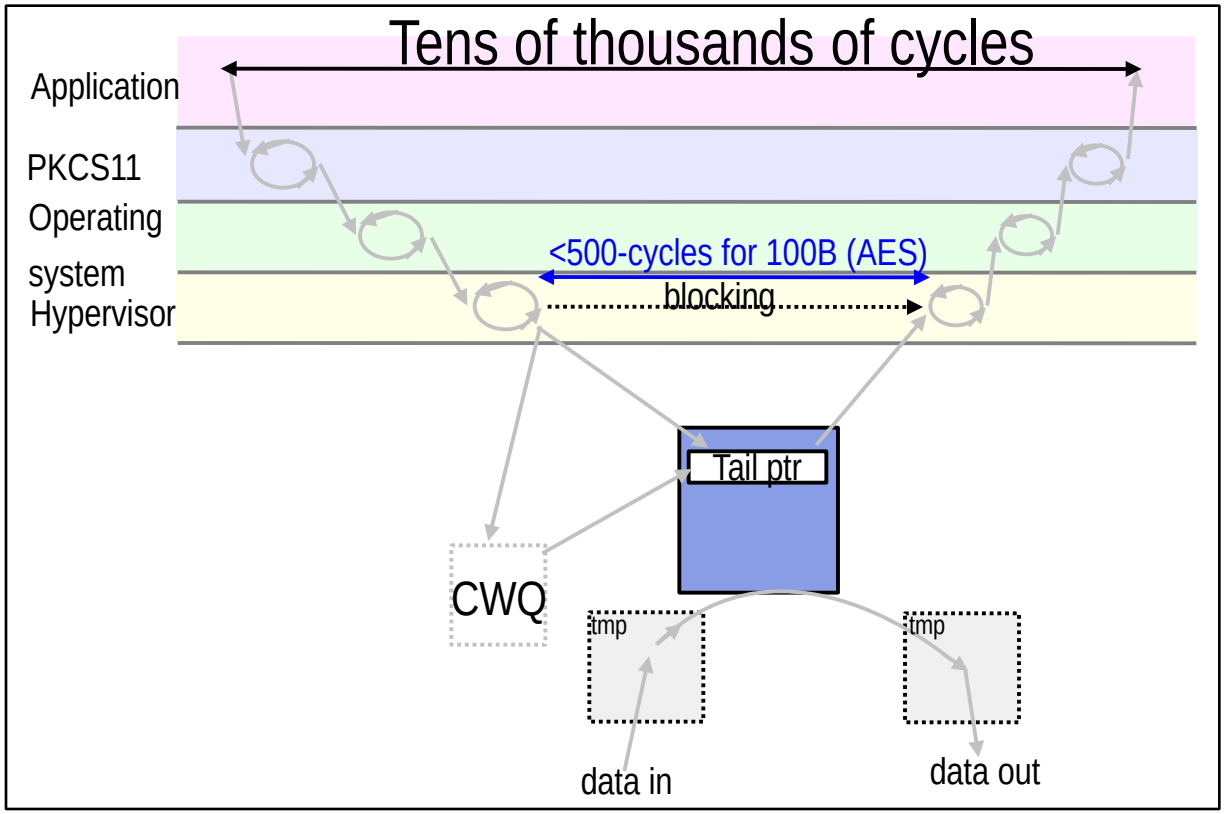
Cryptographic framework overheads

- Access to the accelerators controlled by Solaris cryptographic framework
- SW stack traversal adds significant overhead to offloads
- Data copying often required due to accelerators use of physical addresses
- Basic SW architecture mirrored in many other cryptographic frameworks
  - e.g. Open Cryptographic Framework (OCF)
- Classic frameworks introduce significant software overheads to accelerator offloads
  - OK for long latency public-key operations
  - Not as obvious for offchip accelerator cards
- Most problematic for server-class processors
  - Embedded security processors typically run with simple executive



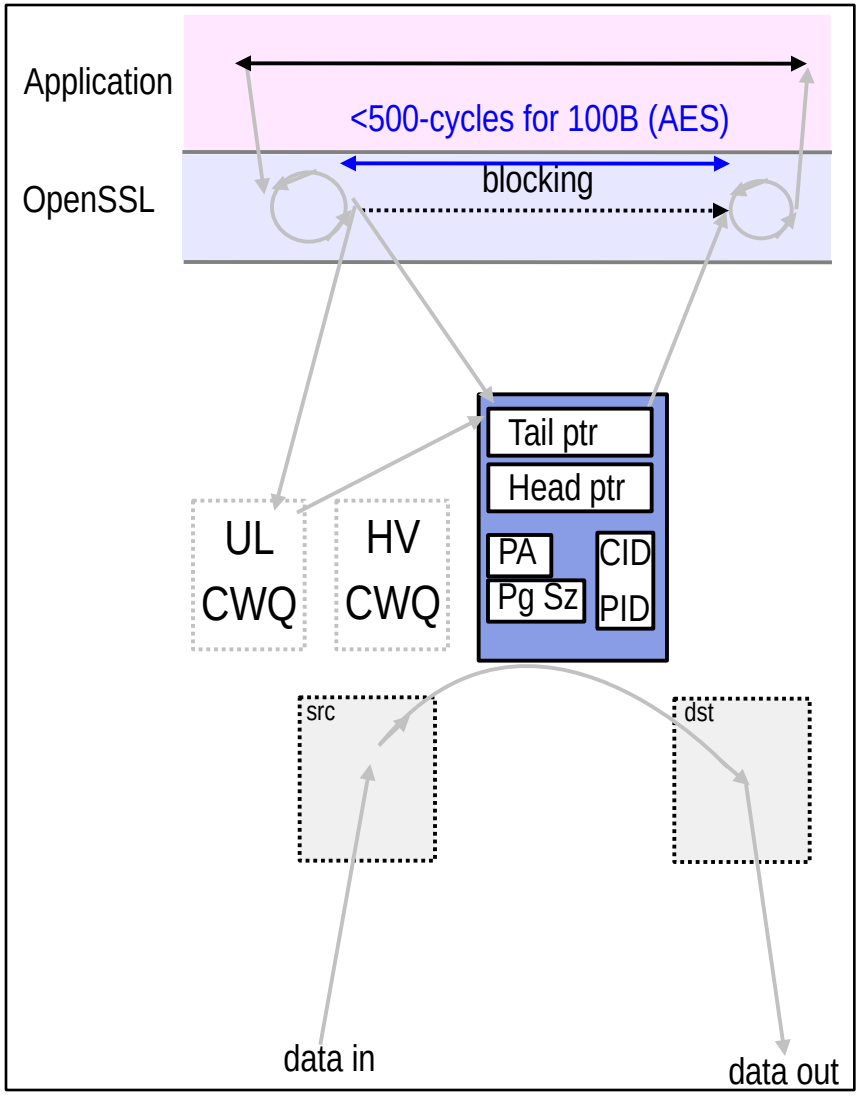
RF accelerator fast-path - motivation

- For onchip accelerators to be effective, these software overheads must be reduced
  - > Current situation curtailing small packet performance
  - > This requirement is not crypto specific and is mirrored by most acceleratable operations



RF accelerator fast-path - overview

- OS/HV traditionally involved on every interaction with the accelerator
- Could enhance the accelerator such that only 1 Hypervisor (HV) interaction is required per session
  - > Only 1st access would require HV "approval"
  - > All subsequent accesses should proceed without HV or OS intervention
- Allowing a non-priv application to directly access the accelerators would virtually eliminate the SW overheads



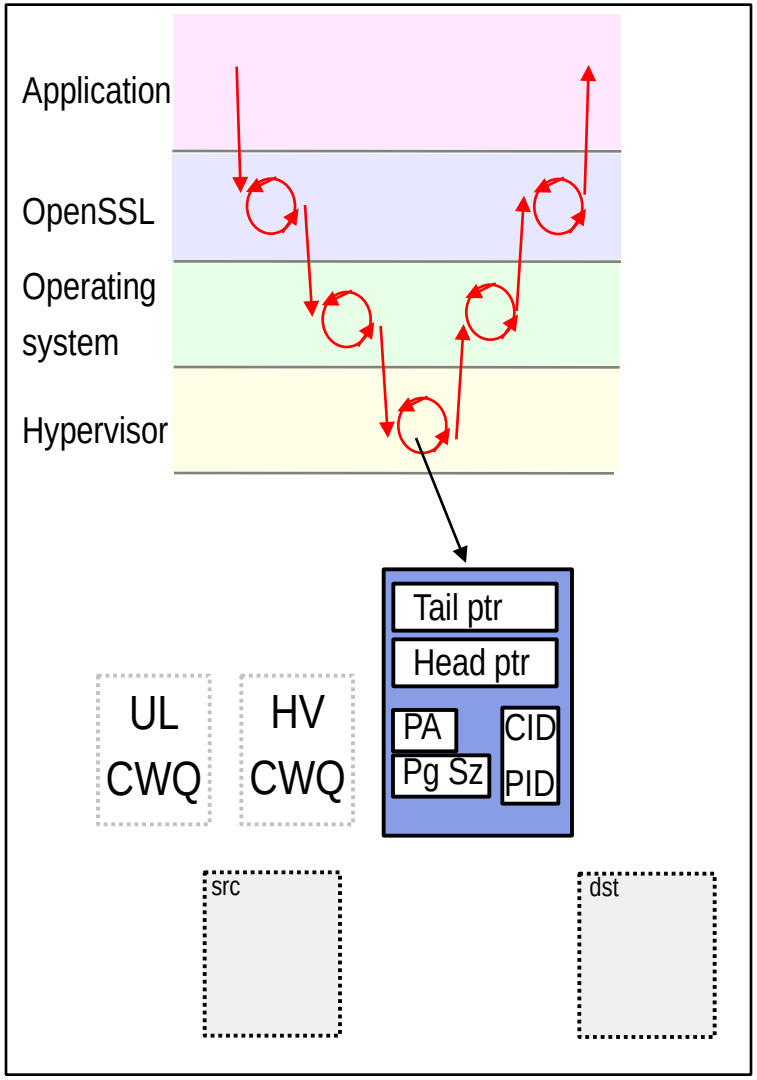
Challenges

- Provide user applications with direct access to a shared resource while ensuring:
  - Security for the user
  - Protection from malicious users
  - Minimum modifications to existing software
  - Flexibility
- User has limited control over their environment
  - Thread can be switched out at ANY time
  - Thread could be moved between cores at ANY time
  - Thread's access to the accelerator could be revoked at ANY time
- Accelerators operate on physical addresses
  - Application needs to pass pointers to accelerator without opportunity for abuse
  - OS can page-out application data at ANY time
- Multiple threads within a single user process may need concurrent access to the accelerators
- Multiple user processes may want concurrent access to the accelerators

RF accelerator fast-path - details

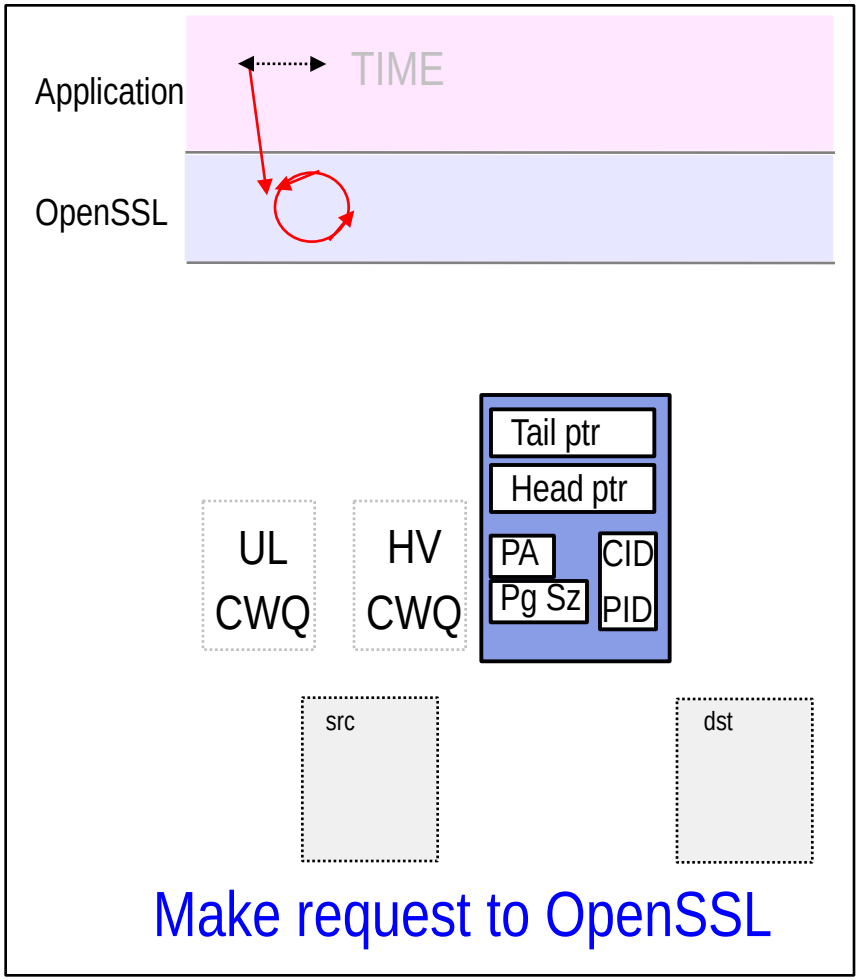
- Initial mediation between the accelerator and user application is performed by the Hypervisor (HV)
- Correct behaviour is subsequently enforced by the accelerator hardware
  - > User requests are uniquely tagged by the HW to allow the accelerators to identify authorized users
  - > Standard address space protections leveraged to secure data
  - > Requests from unauthorized users are ignored by the accelerators
  - > CWQ and objects to be processed are constrained to known pinned pages for which the accelerator has the physical address (TLB on a budget :-)
- Key requirement was to minimize modifications to T2 accelerator
- Augmented existing accelerator with:
  1) Space for limited virtual to physical address translations & page size info
  2) Storage for authorized process partition ID (PID) and context ID (CID) information
  3) Non-priv equivalents for subset of accelerator commands

RF accelerator initialization



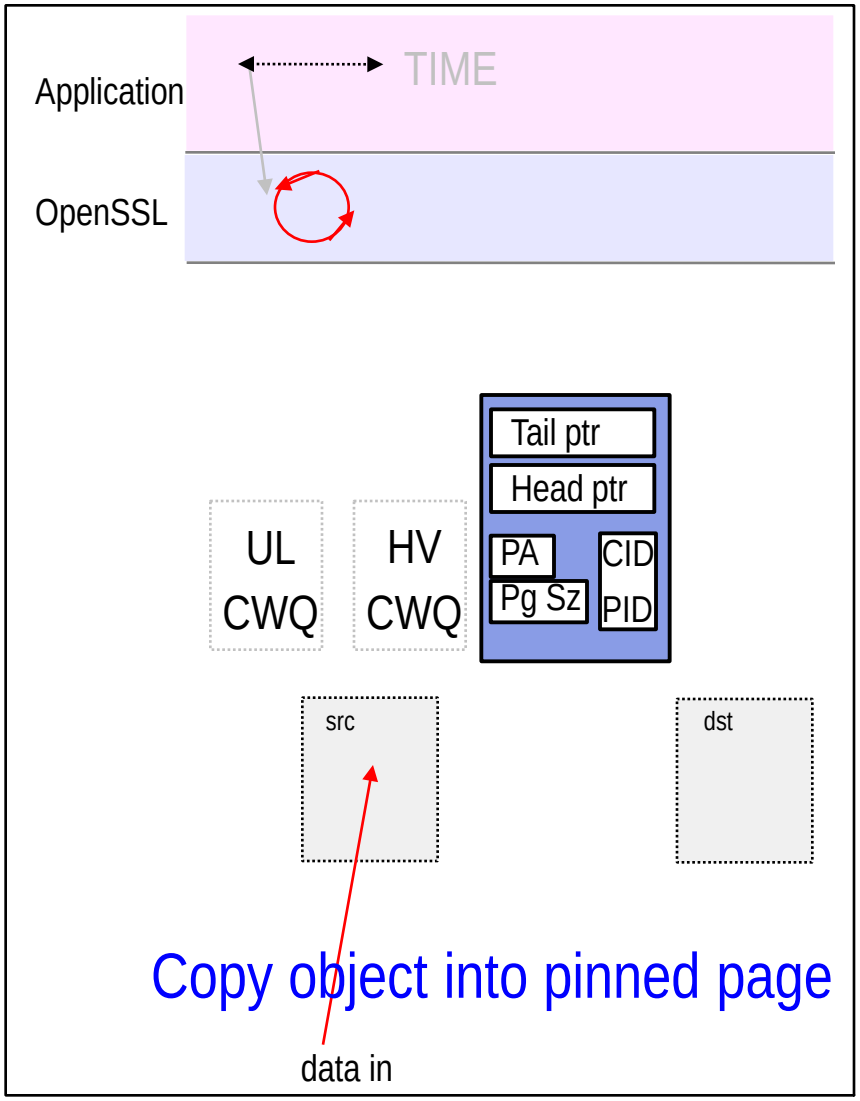
- SW requests direct access to the accelerator from OS/HV
  - > If accelerator is available, HV may grant request
- HV provides accelerator with
  1) CID/TID information of requesting process (uniquely identifies requesting process)
  2) Physical address of buffer in which application will place data to be processed
  3) Physical address of application's control-word queue (CWQ)
- HV provides application with virtual address of buffer and CWQ
- Only required once per process
  - > Occurs 1st time any thread wants to obtain direct access to an accelerator

RF user-privileged operation (1/4)



- New accelerator interface not exposed directly to users
- By leveraging existing APIs user apps don't require recoding
  1) Could also utilize other libraries e.g. PKCS11, NSS

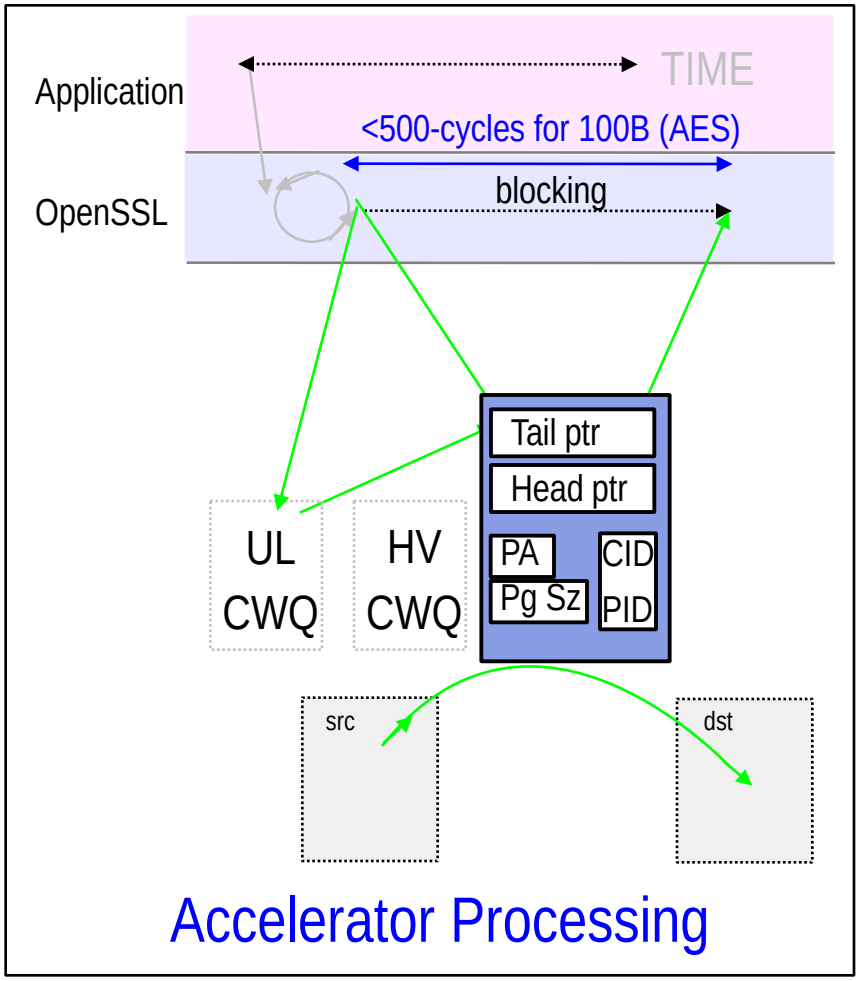
RF user-privileged operation (2/4)



- SW/HW interaction is designed such that;
  - > HV can remove access to the accelerator at any time
  - > SW elegantly recovers from accelerator removal or inter-core migration during programming
  - > SW ensure MT processes safely share the CWQ

- > Objects to be processed are placed in the src/dst page
- > Accelerator will refuse to process objects not contained in the src page (preventing access violations)
- > By forcing communication via this page, the VA->PA conversion problems are avoided – the accelerator has a translation for this page

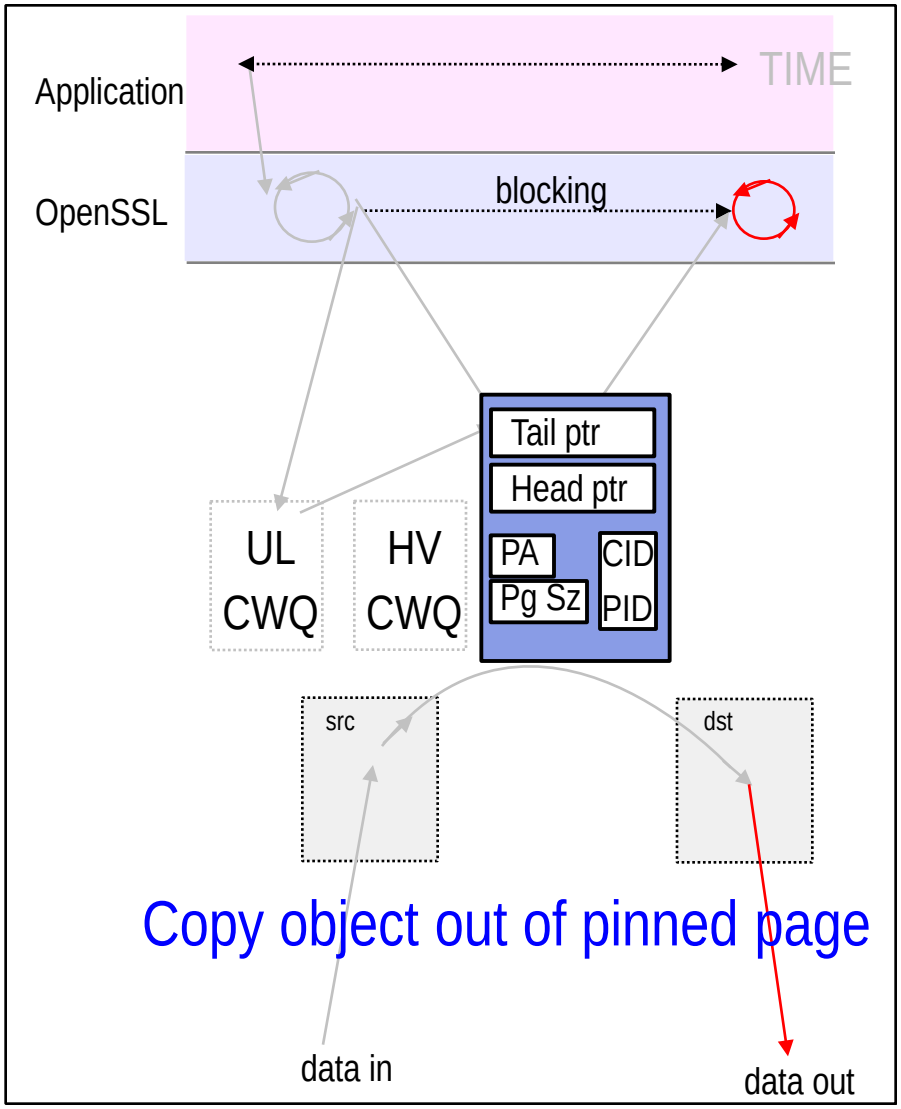
RF user-privileged operation (3/4)



- Application interacts directly with the accelerator
  - > Inserts control-word in CWQ
  - > Updates accelerator's CWQ pointer (to reflect new entry) via special store instructions
  - > Application queries accelerator using special loads to determine successful completion

- > src/dst page is pinned by the OS, preventing the page from being paged out while accelerator is operating
- > Removes requirement for accelerator to snoop all demaps

RF user-privileged operation (4/4)



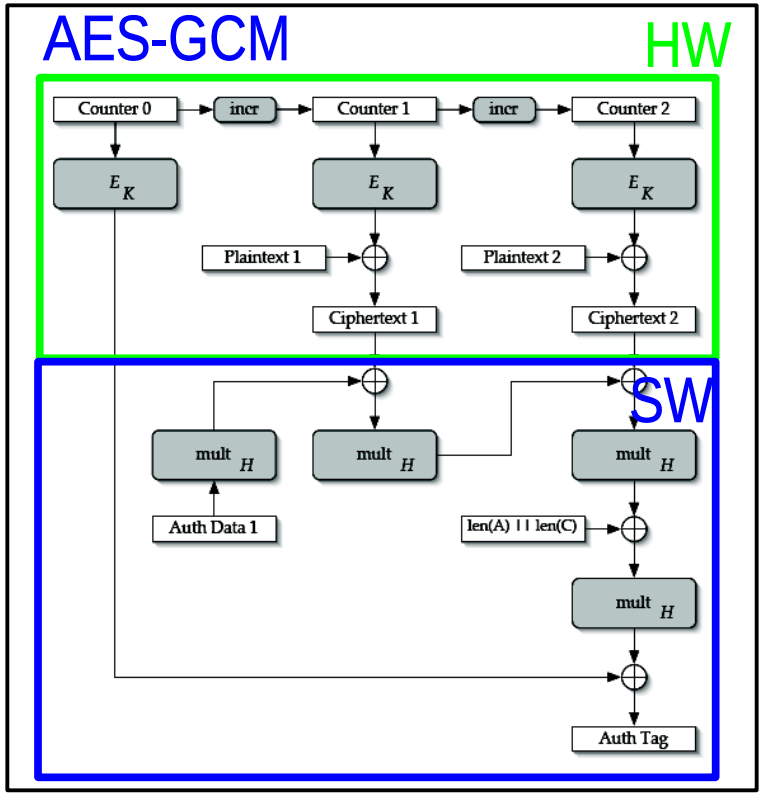
- In-place transforms are not permitted
  - > Allows accelerator operations to be aborted at ANY time
- Cost of moving data in and out of the pinned page is trivial for small objects
  - > Copy can be eliminated with careful object placement

RF accelerator fast-path - performance

- RF 'fast-path' improves application-level small packet performance by up to 30X (compared to T2)
  - Now just a handful of stores required to program the accelerator
- Allows userland applications to obtain close to HW peak performance for all packet sizes
- Careful application integration can eliminate need for data copying
  - Area/complexity/inheritance constraints prevented a more elegant HW solution
- Fast-path interface can be wrapped in OpenSSL or JCE to allow existing applications to benefit without recoding/recompilation
- Binding can be performed with per- accelerator granularity
  - For cost-saving only one control queue per accelerator
  - Minimal overheads associated with re-targeting control queue

RF support for new chaining modes

- Many newly defined authenticated encryption algorithms e.g. AES-GCM
- RF splits computation between HW and SW
  - Faster than pure ISA-based crypto approach
- For example, AES-GCM:
  - SW performs GHASH computation
    - Efficient with XMULX/XMULXHI instructions
    - Reduces instruction count by about 8X
  - Accelerator performs AES-CTR
- SW can keep pace with HW
- Flexible approach; can readily handle future modes
  - Overcomes notion of inflexibility of discrete accelerators



RF support for Telco ciphers

- Kasumi is used for encryption and authentication in 3GPP
- Reworked SW implementation on T2
  - Narrow, threaded cores alter compute/memory trade-offs
  - Merging several small lookups tables can be beneficial even though it may make tables too large for level-1 caches
  - Reducing compute improved scaling and aggregate performance

```
nine = (u16)(in>>7);
seven = (u16)(in&0x7F);
nine = (u16)(S9[nine] ^ seven);
seven = (u16)(S7[seven] ^ (nine & 0x7F));
seven ^= (subkey>>9);
nine ^= (subkey&0x1FF);
nine = (u16)(S9[nine] ^ seven);
seven = (u16)(S7[seven] ^ (nine & 0x7F));
in = (u16)((seven<<9) + nine);
return( in );
```

```
t0 = LT0[in];
t0 = t0 ^ subkey;
in = LT1[t0];
return(in);
```

2 level-1 cache resident tables [128 & 512 elements]
Compute dominates
Poor scaling on SMT cores

2 level-2 cache resident tables [65536 elements each]
Limited compute
Great scaling on SMT cores
Comparable single-thread performance
Greatly improved aggregate MT performance

- RF HW performance still very advantageous

ISA-based crypto acceleration

Given discrete accelerator complexity why not adopt an instruction based approach to cryptographic acceleration?

- Limited pipeline resources on CMT processors
  - Highly shared pipelines easily monopolized by crypto operations
  - Leaves cores for processing to which they are well suited
  - On order of 60X reduction in pipeline utilization for 1KB object (discrete versus ISA)
- Typically higher performance on a per cycle basis if don't have to partition the computation into instructions
  - Important for lower frequency, power efficient CMT processors
- Discrete accelerators typically more power efficient method for performing crypto operations
- Discrete accelerators can help minimize cache pollution
- Not all crypto operations cleanly sub-divide into manageable crypto instructions

Summary

- RF continues UltraSPARC CMT tradition of providing on-chip accelerators
- RF includes Sun's 3rd generation on-chip security accelerator
- RF's accelerator introduces
  - Additional ciphers, chaining modes and secure hashes
  - Non-priv fast-path to accelerators
- Fast-path eliminates vast majority of overheads associated with offloads
  - Allows direct interaction between non-priv applications and the accelerators
  - Improves small object performance by up to 30X
- RF provides additional non-priv crypto instructions to help accelerate authenticated-encryption operations
- RF builds on the successes of the UltraSPARC T2 and significantly expands the application space which can benefit from the accelerators

Acknowledgements

Farnad Sajjadian

Chris Olson

Sanjay Patel

Greg Grohoski

Stephen Phillips

Thank you ...

Lawrence.spracklen@sun.com