STAN – Current State and Future Work

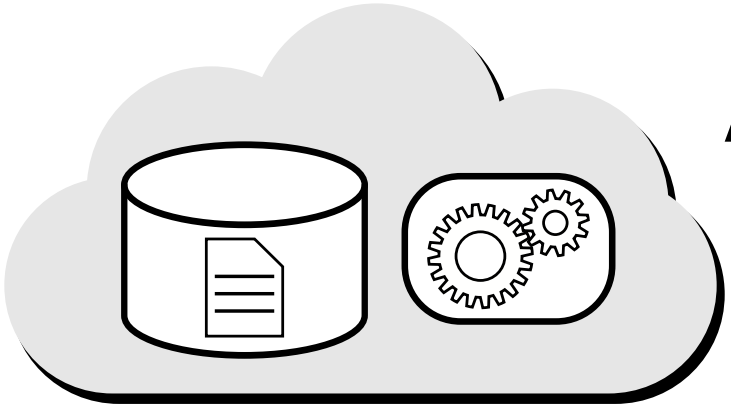
Scalable Hardware-Aided Trusted Data Management

Sebastian Krieter, Nico Weichbrodt, 2018-09-17

Scalable Data Management

- Collaboration between TU Braunschweig and HS Harz
- Shift from **on-premise** to **cloud**
  - Economy of scale, …

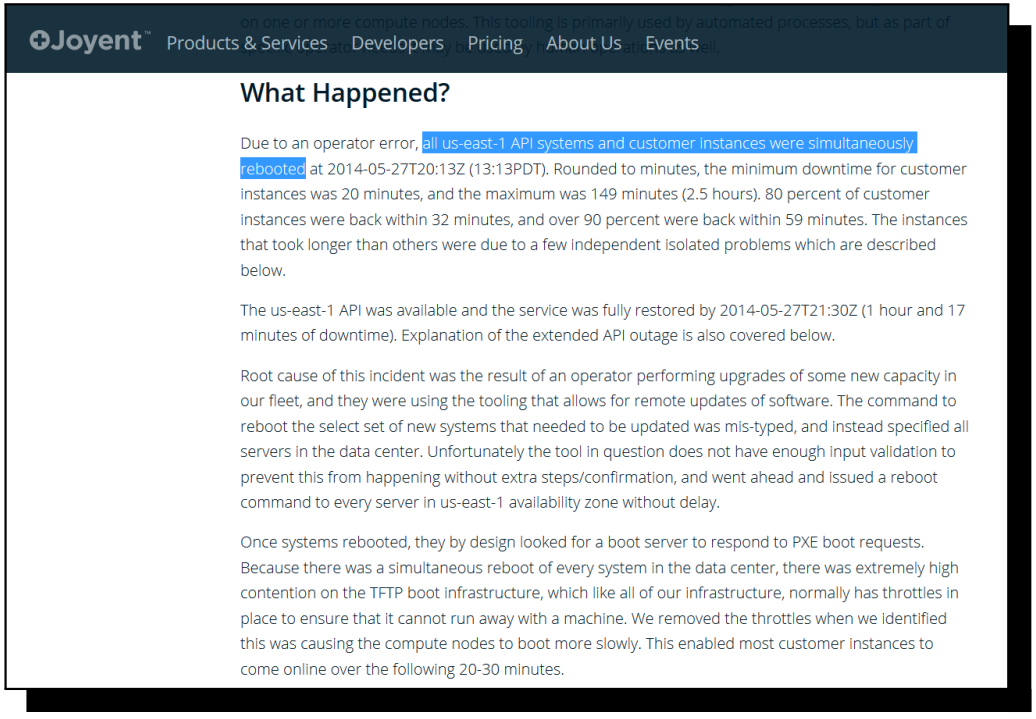
Scalability



Availability

Computational Outsourcing

Security Issues in the Cloud



Joyent Products & Services Developers Pricing About Us Events

What Happened?

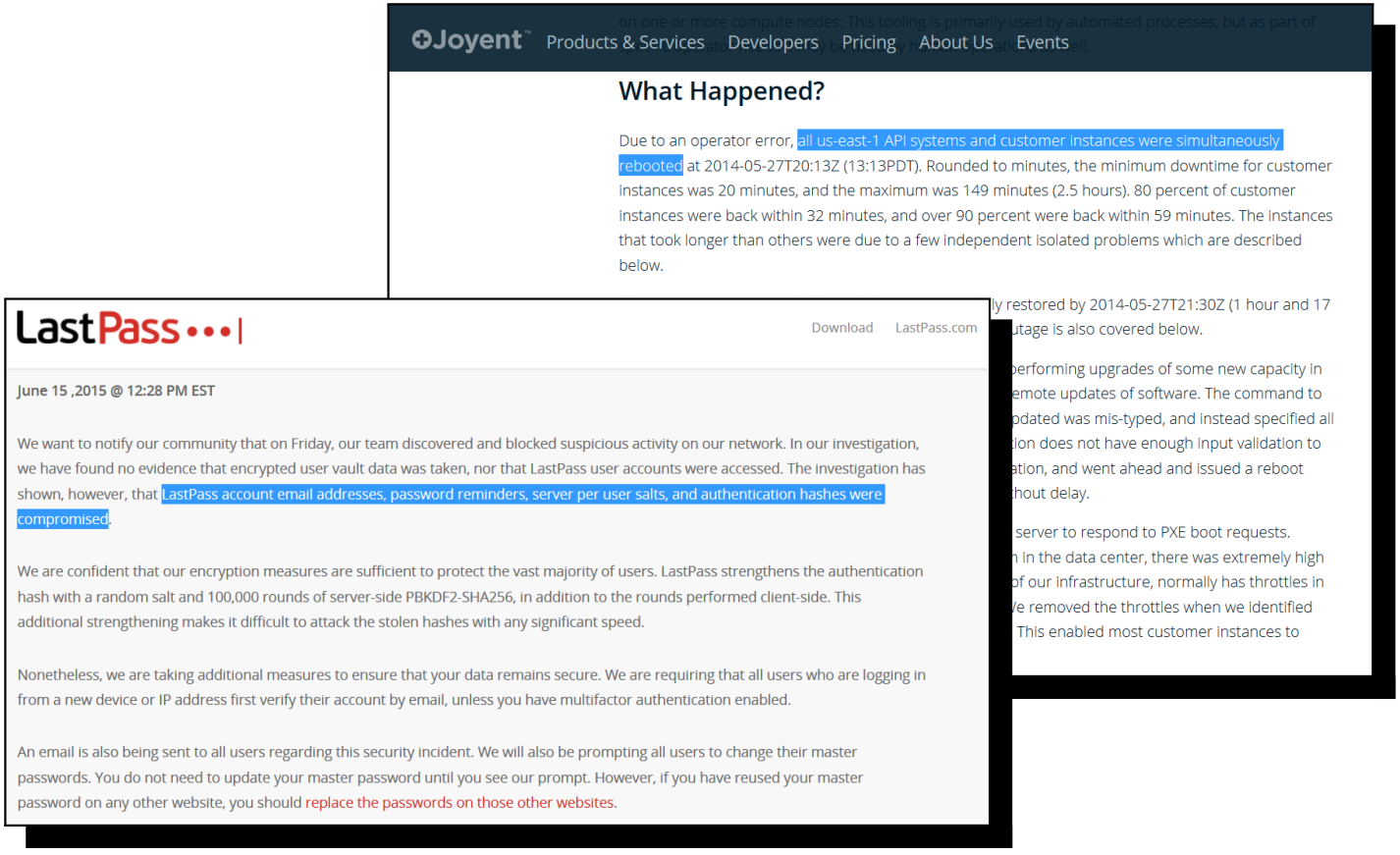
Due to an operator error, all us-east-1 API systems and customer instances were simultaneously rebooted at 2014-05-27T20:13Z (13:13PDT). Rounded to minutes, the minimum downtime for customer instances was 20 minutes, and the maximum was 149 minutes (2.5 hours). 80 percent of customer instances were back within 32 minutes, and over 90 percent were back within 59 minutes. The instances that took longer than others were due to a few independent isolated problems which are described below.

The us-east-1 API was available and the service was fully restored by 2014-05-27T21:30Z (1 hour and 17 minutes of downtime). Explanation of the extended API outage is also covered below.

Root cause of this incident was the result of an operator performing upgrades of some new capacity in our fleet, and they were using the tooling that allows for remote updates of software. The command to reboot the select set of new systems that needed to be updated was mis-typed, and instead specified all servers in the data center. Unfortunately the tool in question does not have enough input validation to prevent this from happening without extra steps/confirmation, and went ahead and issued a reboot command to every server in us-east-1 availability zone without delay.

Once systems rebooted, they by design looked for a boot server to respond to PXE boot requests. Because there was a simultaneous reboot of every system in the data center, there was extremely high contention on the TFTP boot infrastructure, which like all of our infrastructure, normally has throttles in place to ensure that it cannot run away with a machine. We removed the throttles when we identified this was causing the compute nodes to boot more slowly. This enabled most customer instances to come online over the following 20-30 minutes.

Security Issues in the Cloud



Joyent Products & Services Developers Pricing About Us Events

What Happened?

Due to an operator error, all us-east-1 API systems and customer instances were simultaneously rebooted at 2014-05-27T20:13Z (13:13PDT). Rounded to minutes, the minimum downtime for customer instances was 20 minutes, and the maximum was 149 minutes (2.5 hours). 80 percent of customer instances were back within 32 minutes, and over 90 percent were back within 59 minutes. The instances that took longer than others were due to a few independent isolated problems which are described below.

LastPass Download LastPass.com

June 15 ,2015 @ 12:28 PM EST

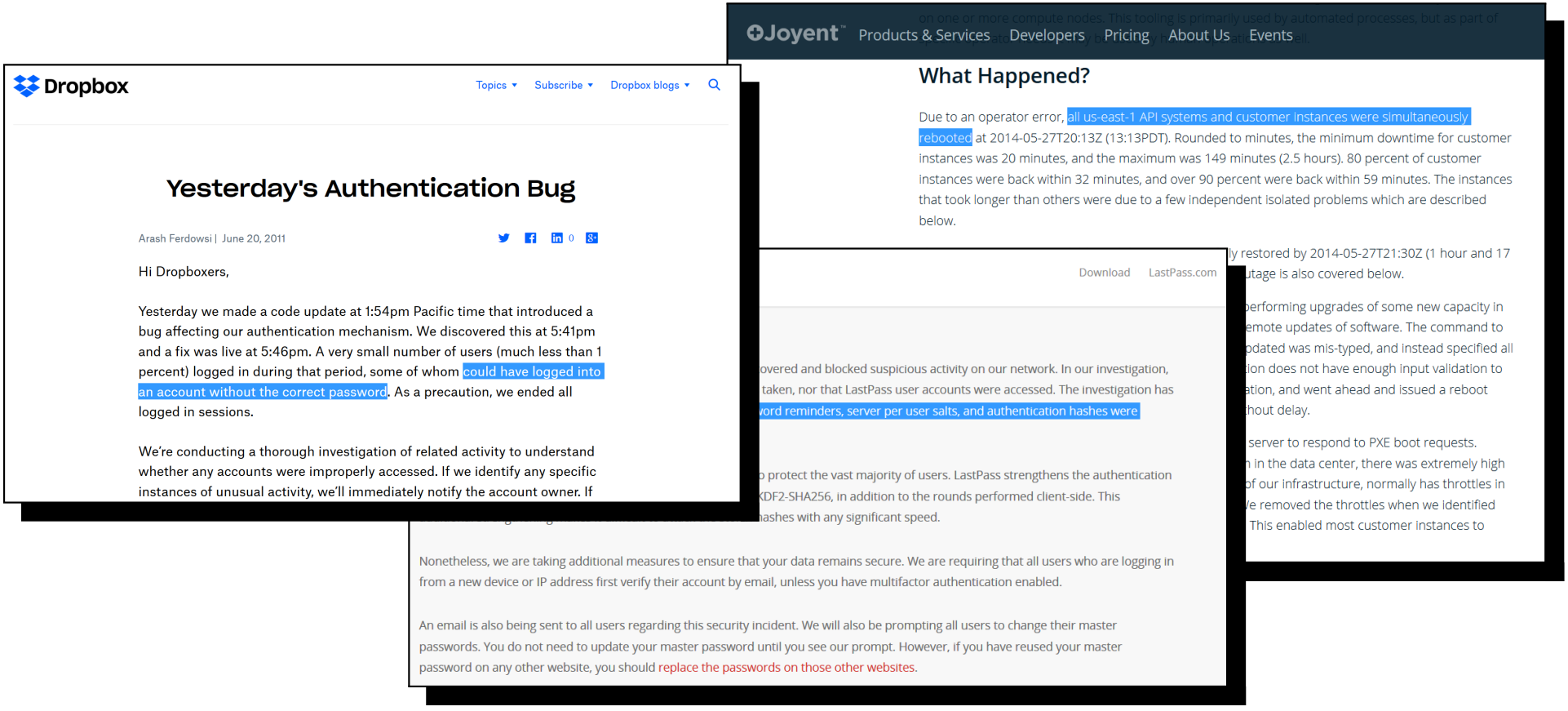
We want to notify our community that on Friday, our team discovered and blocked suspicious activity on our network. In our investigation, we have found no evidence that encrypted user vault data was taken, nor that LastPass user accounts were accessed. The investigation has shown, however, that LastPass account email addresses, password reminders, server per user salts, and authentication hashes were compromised.

We are confident that our encryption measures are sufficient to protect the vast majority of users. LastPass strengthens the authentication hash with a random salt and 100,000 rounds of server-side PBKDF2-SHA256, in addition to the rounds performed client-side. This additional strengthening makes it difficult to attack the stolen hashes with any significant speed.

Nonetheless, we are taking additional measures to ensure that your data remains secure. We are requiring that all users who are logging in from a new device or IP address first verify their account by email, unless you have multifactor authentication enabled.

An email is also being sent to all users regarding this security incident. We will also be prompting all users to change their master passwords. You do not need to update your master password until you see our prompt. However, if you have reused your master password on any other website, you should replace the passwords on those other websites.

Security Issues in the Cloud



Scalable Hardware-Aided Trusted Data Management

- Cloud is not trustworthy
  - Rogue employees, hypervisor backdoors, ...
- Trusted execution more prevalent
  - ARM TrustZone, AMD SME/SEV, Intel SGX, ...

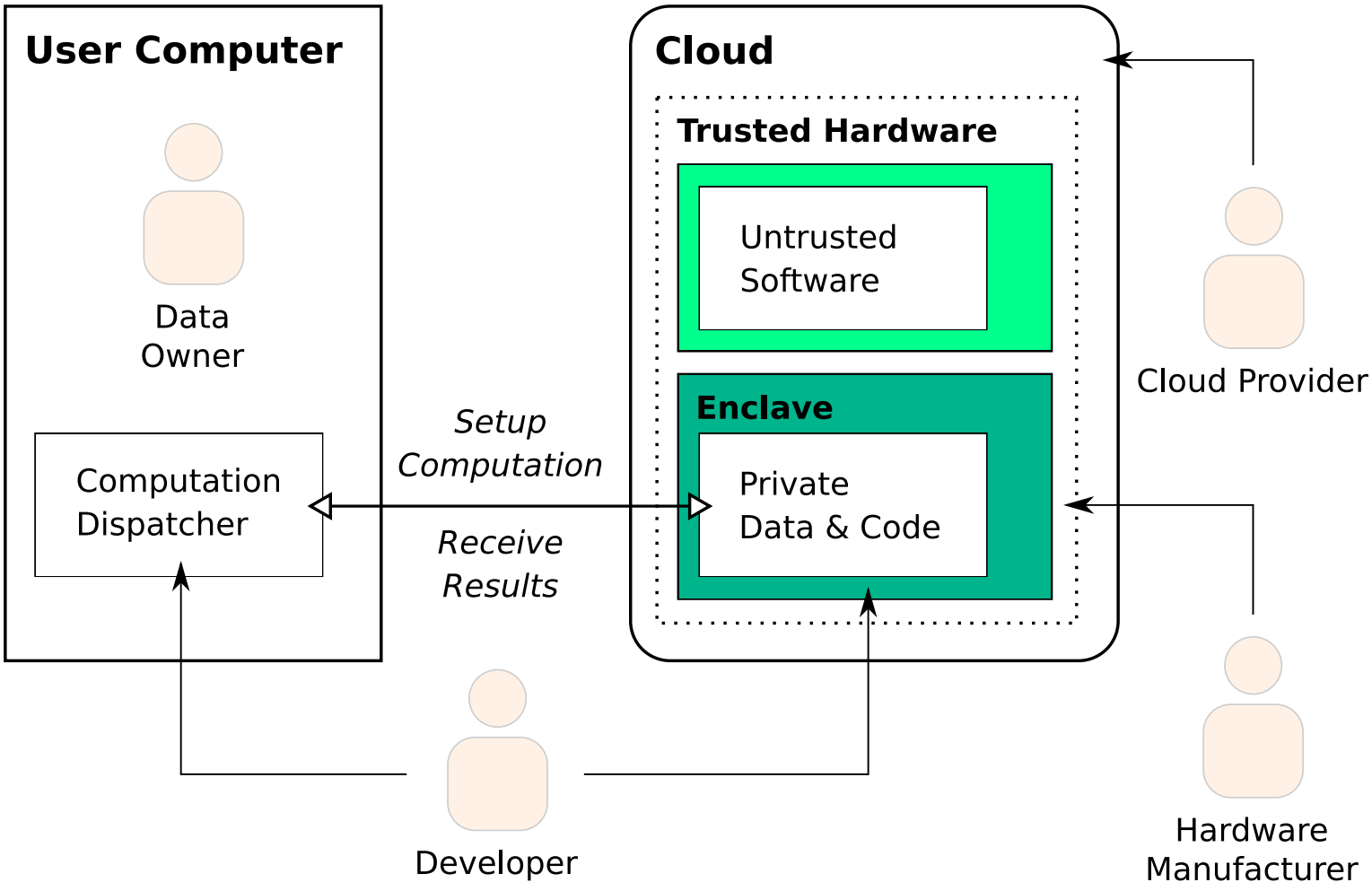
→ Confidentiality and/or Integrity of data

How to combine database systems with trusted execution?

Intel Software Guard Extensions

- Intel SGX security extension for x86
- Add secure memory section to applications called *enclaves*
- Confidentiality & integrity protection

Intel Software Guard Extensions



Intel Software Guard Extensions

- Benefits come at a cost

- Near-native performance in enclave
  - Transition 8,600 - 14,000 cycles (Weisse et al.)
  - ≈5,850 cycles without SDK involvement
  - ≈10,170 cycles with Spectre µCode patches
  - ≈13,100 cycles with Foreshadow (L1 Terminal Fault) µCode patches

Intel Software Guard Extensions

- Benefits come at a cost

- Near-native performance in enclave
  - Transition 8,600 - 14,000 cycles (Weisse et al.)
  - ≈5,850 cycles without SDK involvement
  - ≈10,170 cycles with Spectre µCode patches
  - ≈13,100 cycles with Foreshadow (L1 Terminal Fault) µCode patches
- ≈ 93 MiB memory for all enclaves

  → Paging to main memory, expensive re-encryption & page faults

Challenge

Fit DB code and data into 93 MiB

Paging Overhead

- SGX supports paging from enclave memory to main memory
- Requires re-encryption + integrity checks on load
- Page faults cause enclave transition

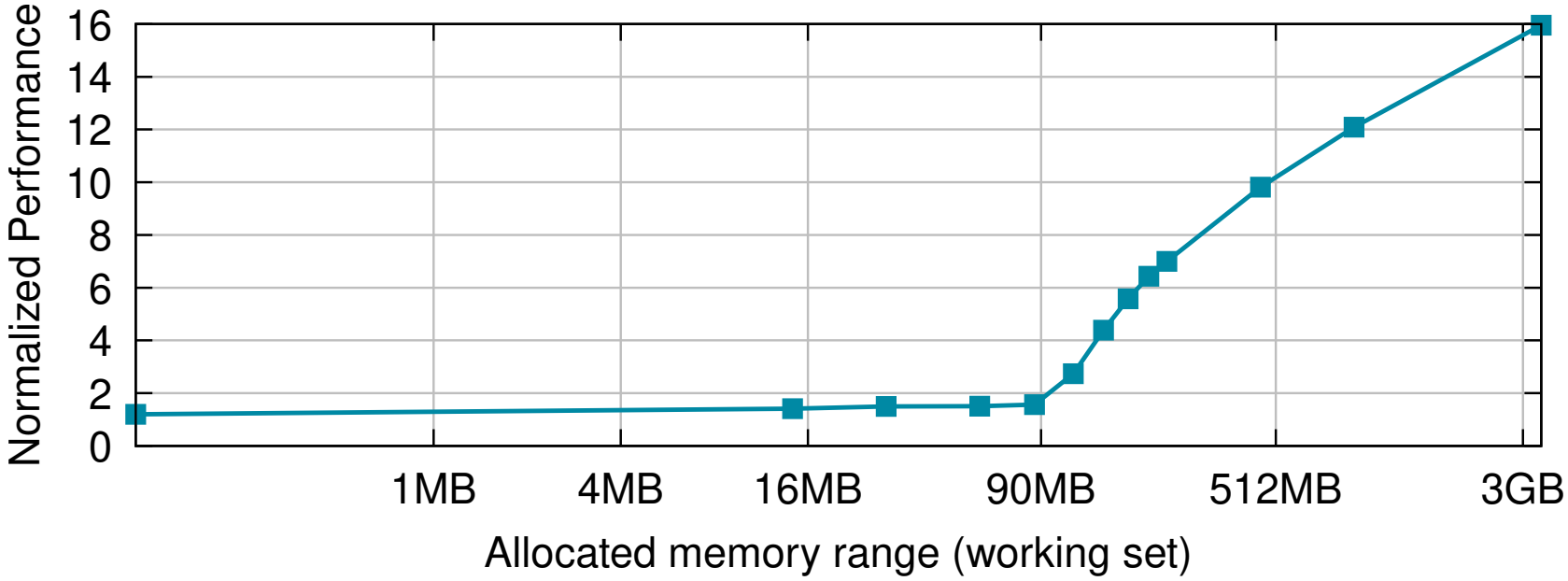
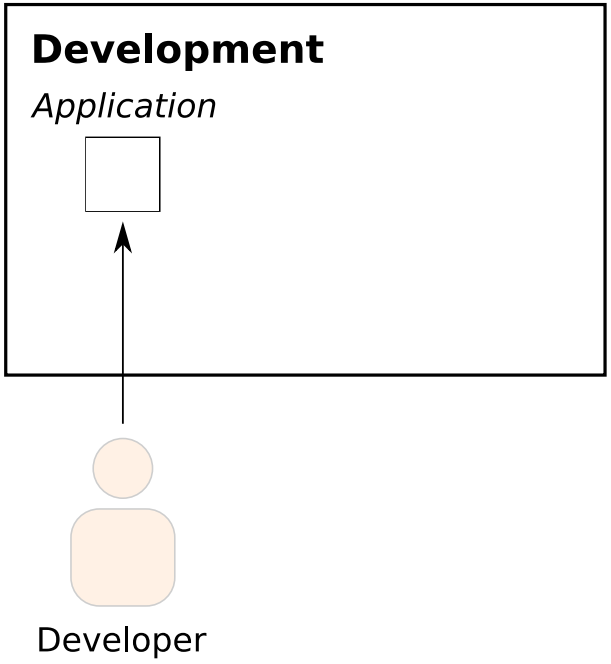
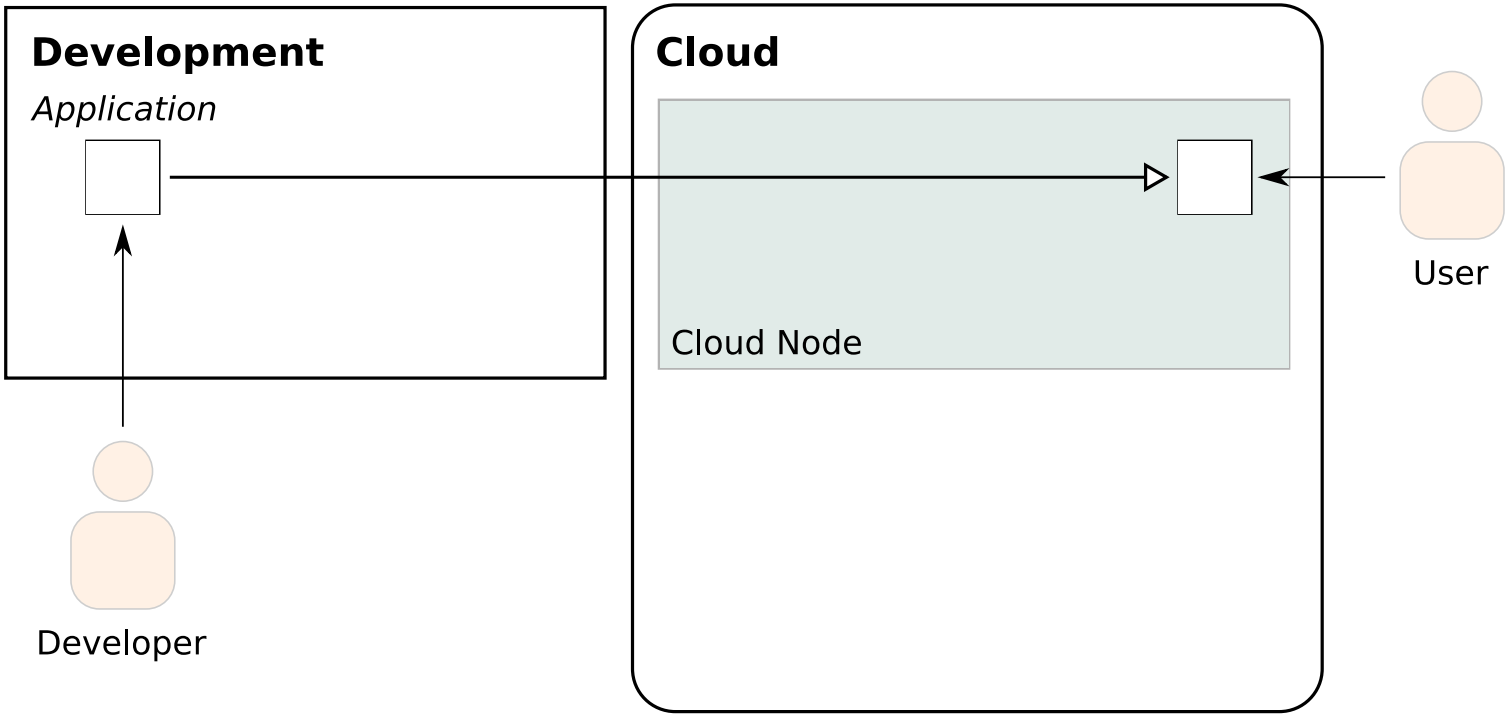


Figure adapted from SecureKeeper, Brenner et al.

Envisioned Approach

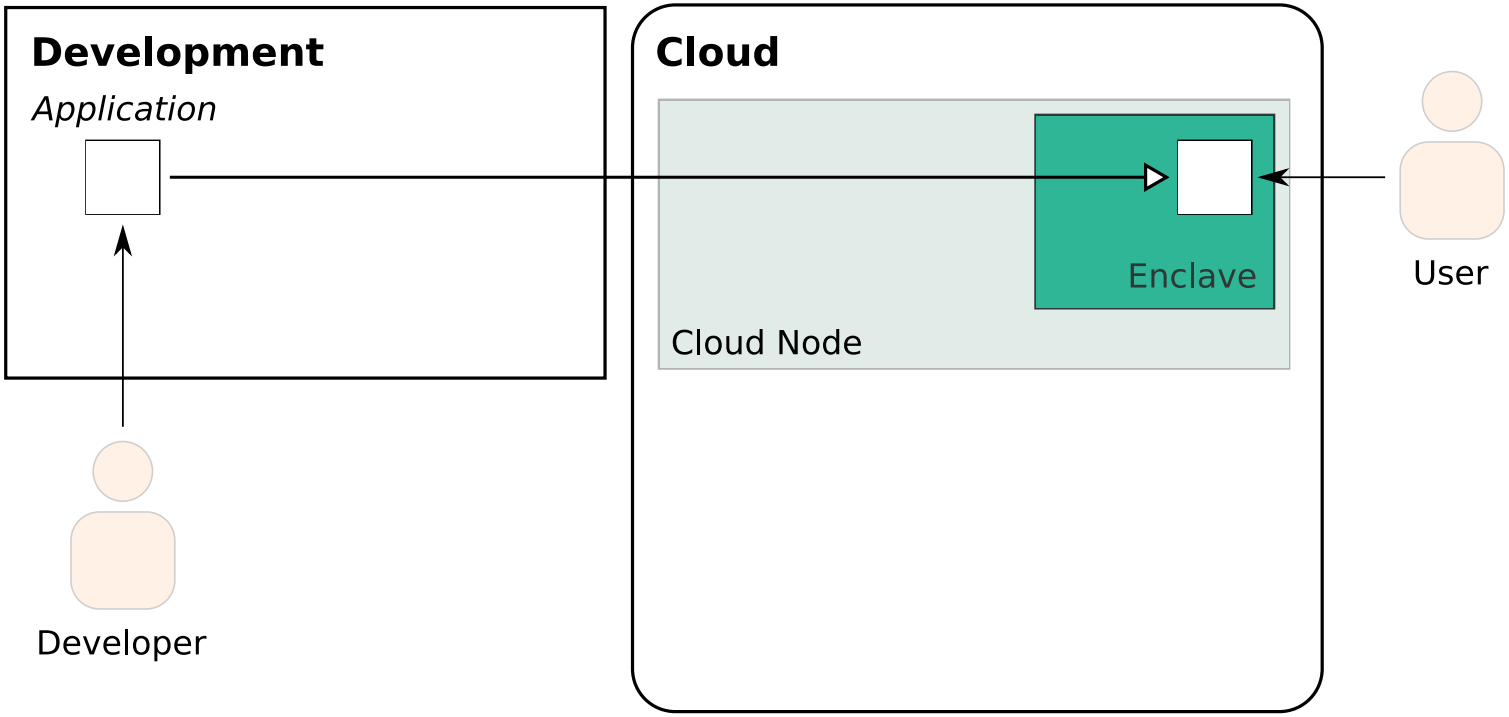


Envisioned Approach



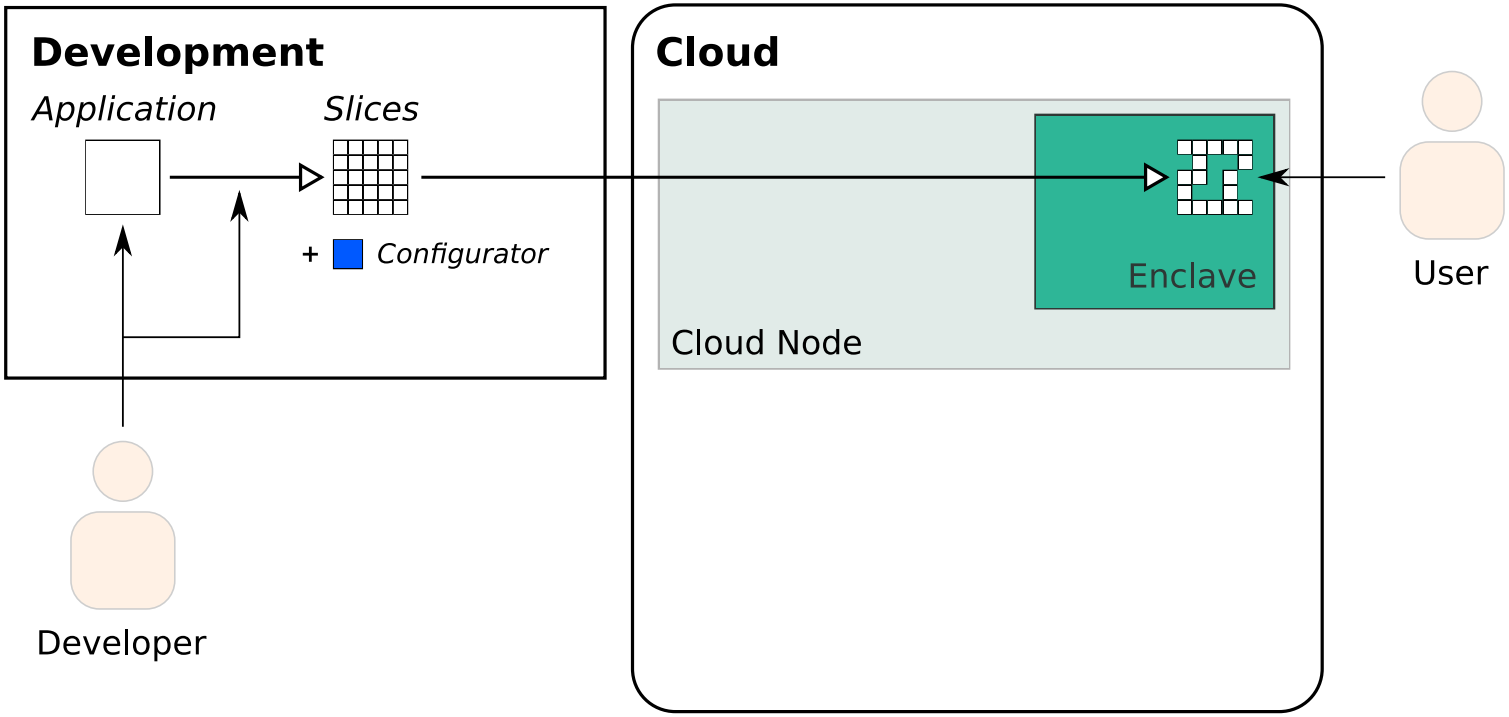
Envisioned Approach

P_1: Protecting the entire code base from unauthorized access



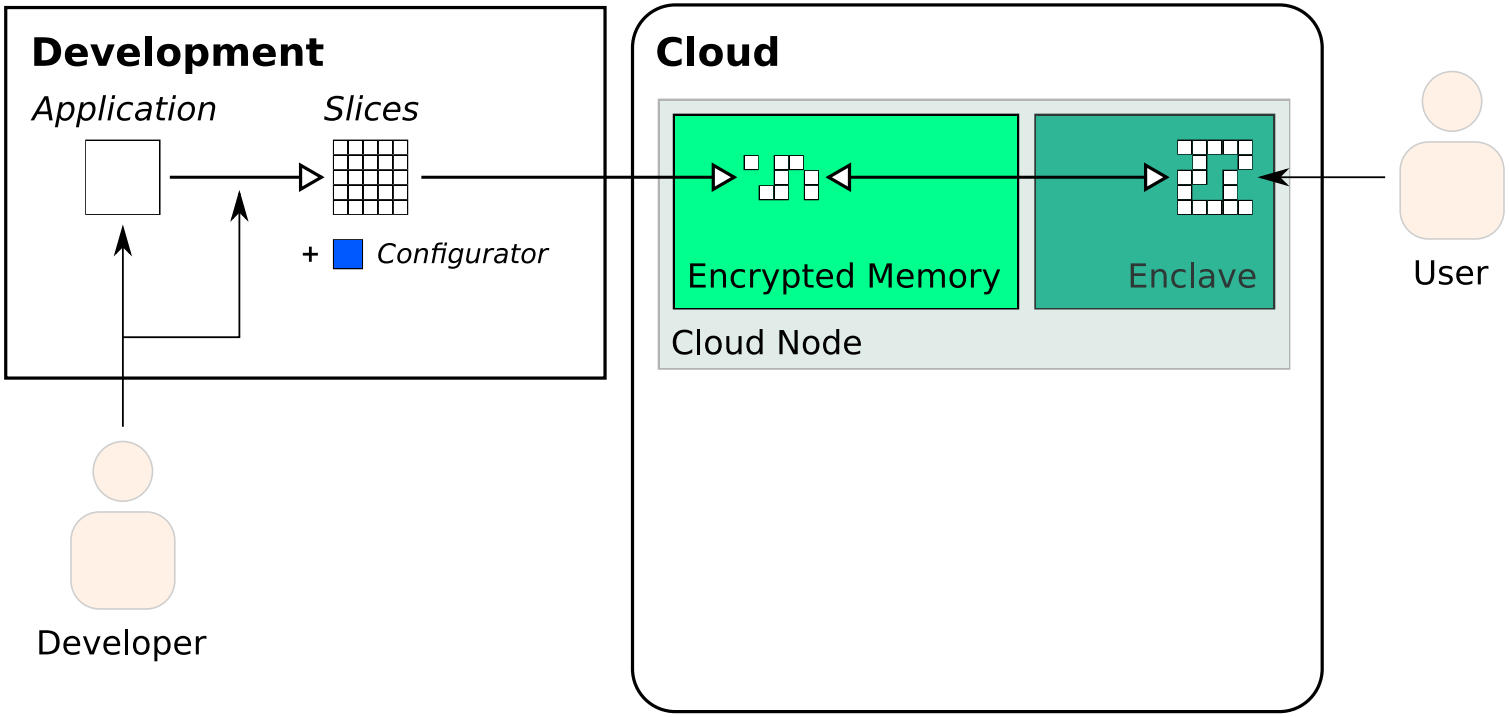
Envisioned Approach

P_2: Running sliced applications within the enclave



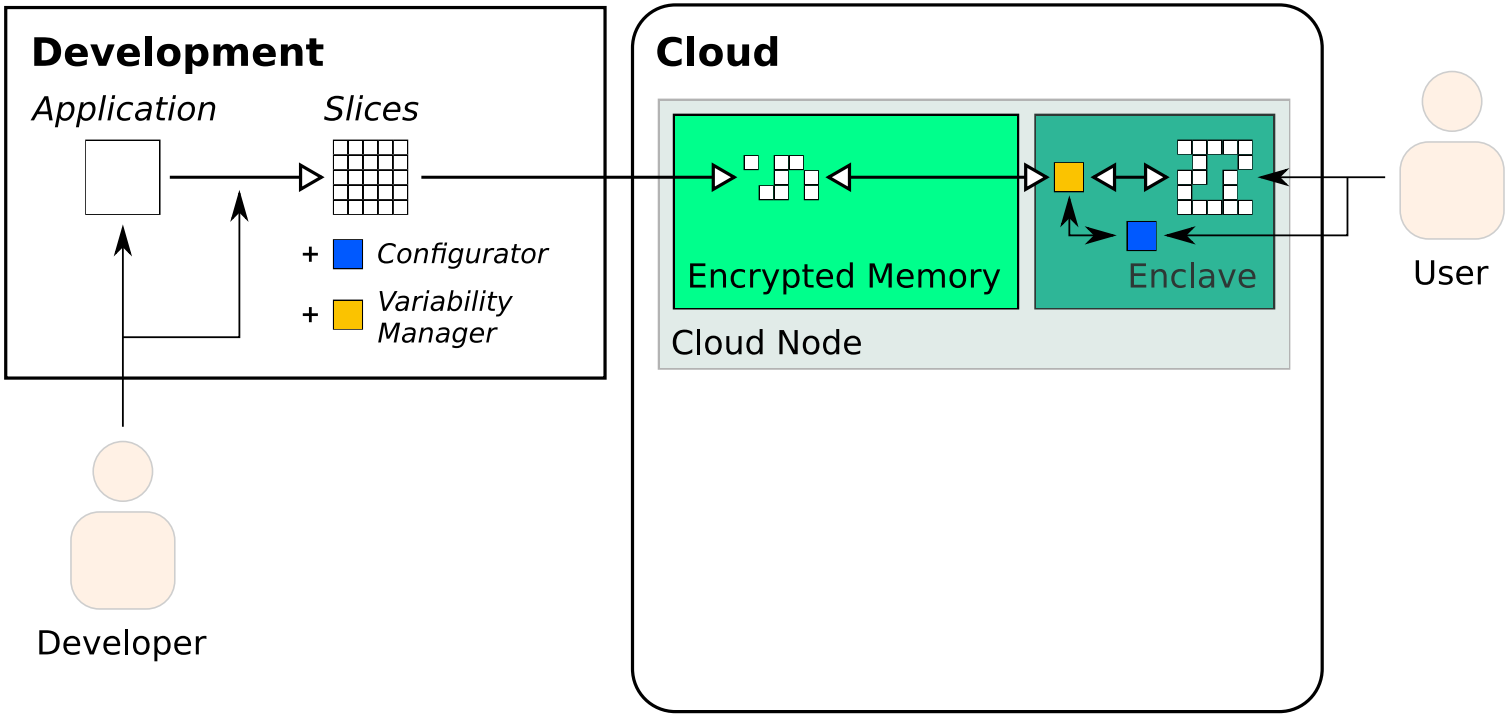
Envisioned Approach

P_3: Enabling dynamic loading of variable application parts



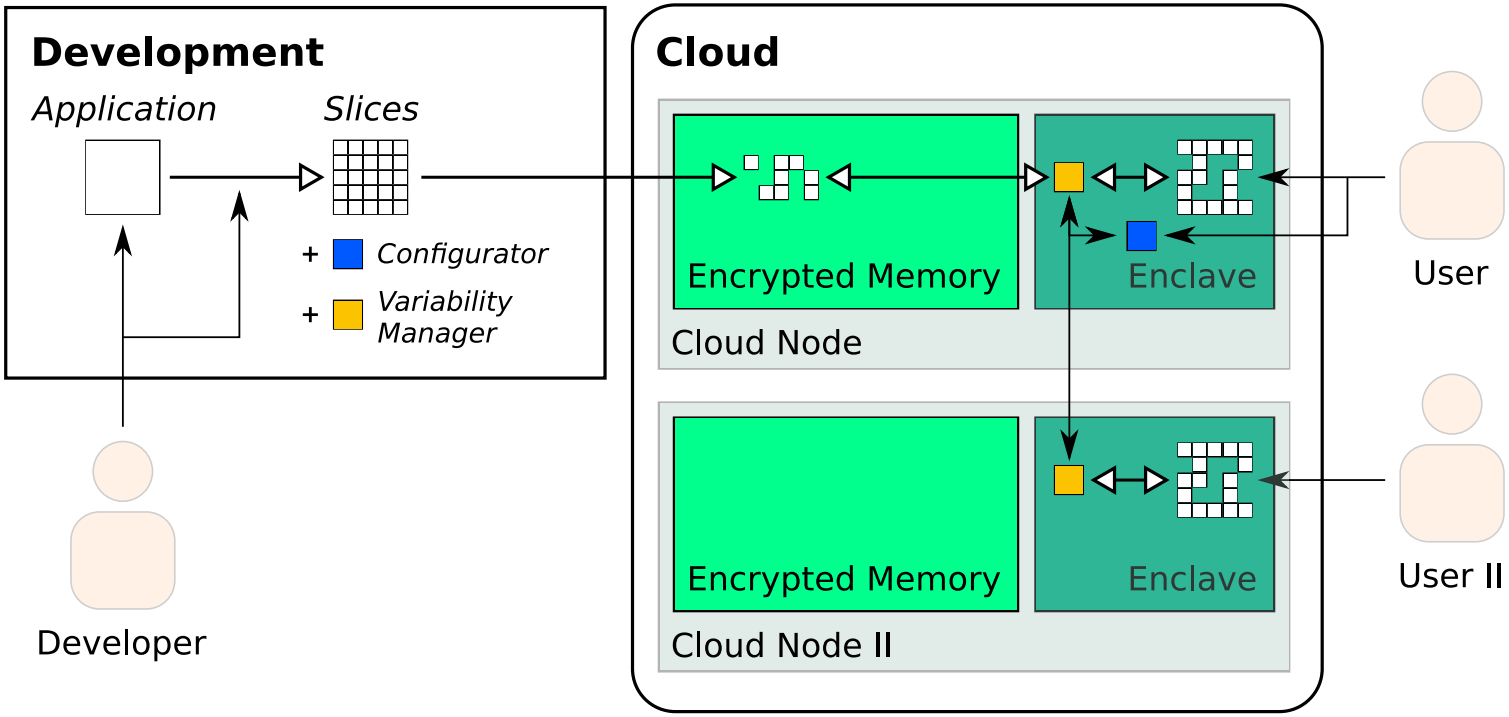
Envisioned Approach

P_4: Supporting self-adaptive reconfiguring of applications



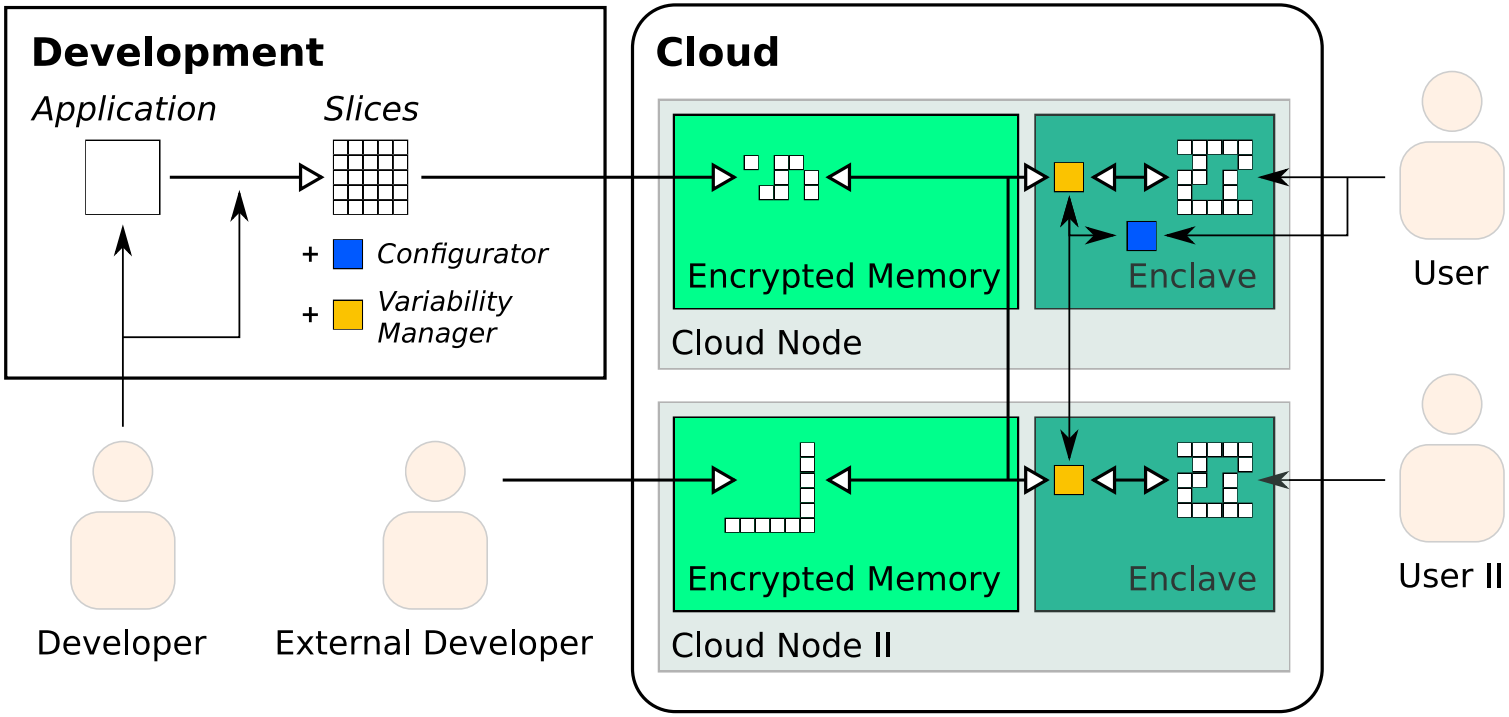
Envisioned Approach

P_5: Utilizing the scalability and availability of cloud computing



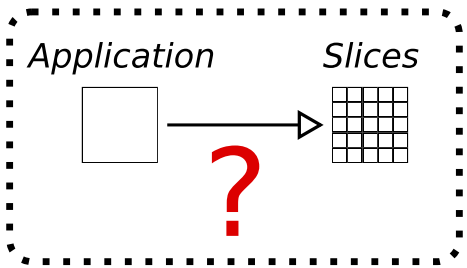
Envisioned Approach

P_6: Including and securing third-party services



Variability in the Enclave

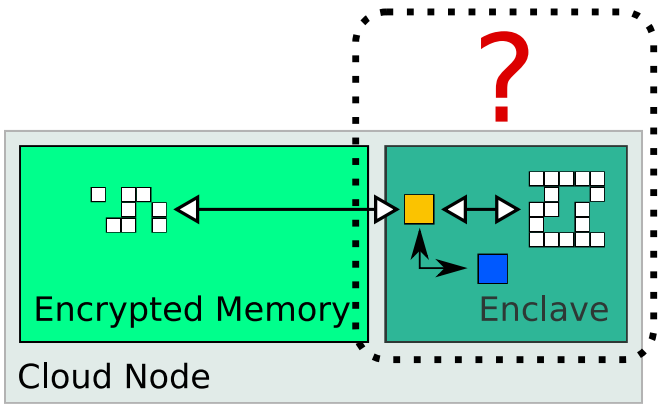
How can we identify meaningful features within an unsliced application?



Self-Adaption at Runtime

What are meaningful heuristics for self-adaptation?

Especially, at which point should features be removed from the enclave?

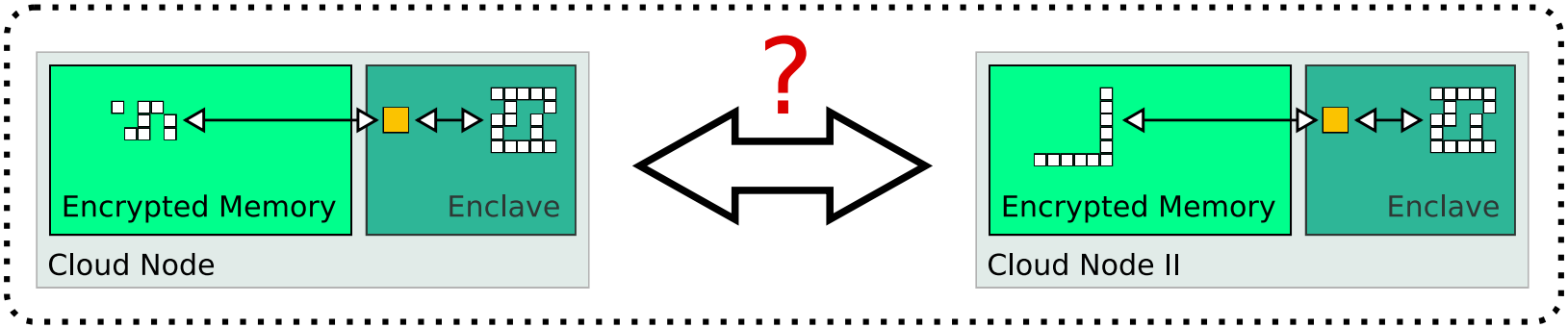


Challenges III

Utilizing the Cloud

How can we efficiently store and transfer the internal configuration of running DSPL among different nodes?

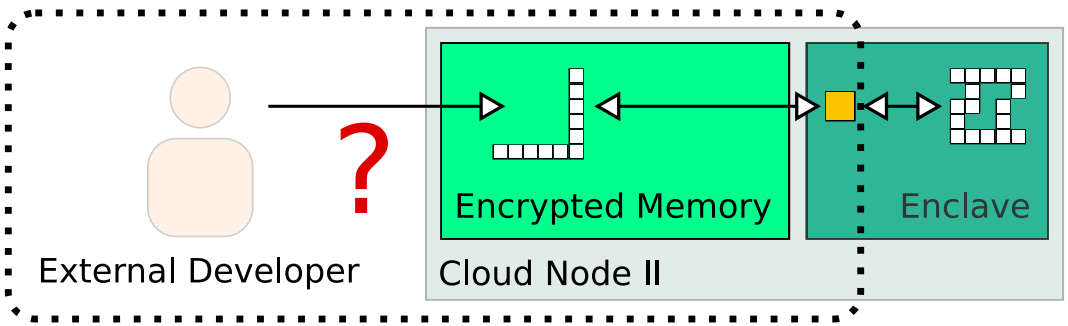
How can we efficiently apply distributed computation in different DSPLs among different nodes?



Integrating Cloud Services

How can we efficiently include third-party services into our DSPL?

How can we efficiently check the integrity of services and their parts?



STANlite[1]

- SQLite database inside an SGX enclave
- Custom-built **virtual memory engine (VME)**
  - Keep encrypted data outside of enclave
- Hand-tailored, no variability, data only

- DB size up to 500 MiB
- Enclave vs native: $\approx$0.6$\times$ ($<$93 MiB), $\approx$0.1$\times$ ($>$93 MiB)
- VME vs native: 0.6-0.5$\times$

$\rightarrow$ Confidentiality + Integrity with 0.5$\times$ native performance

[1] 2018 IEEE International Conference on Cloud Engineering (IC2E)

sgx-perf

- Porting applications manually is time consuming
- Automation is possible: Glamdring (Lind et al., USENIX ATC '17)

But...

Performance is not optimal, excess enclave transitions, ...

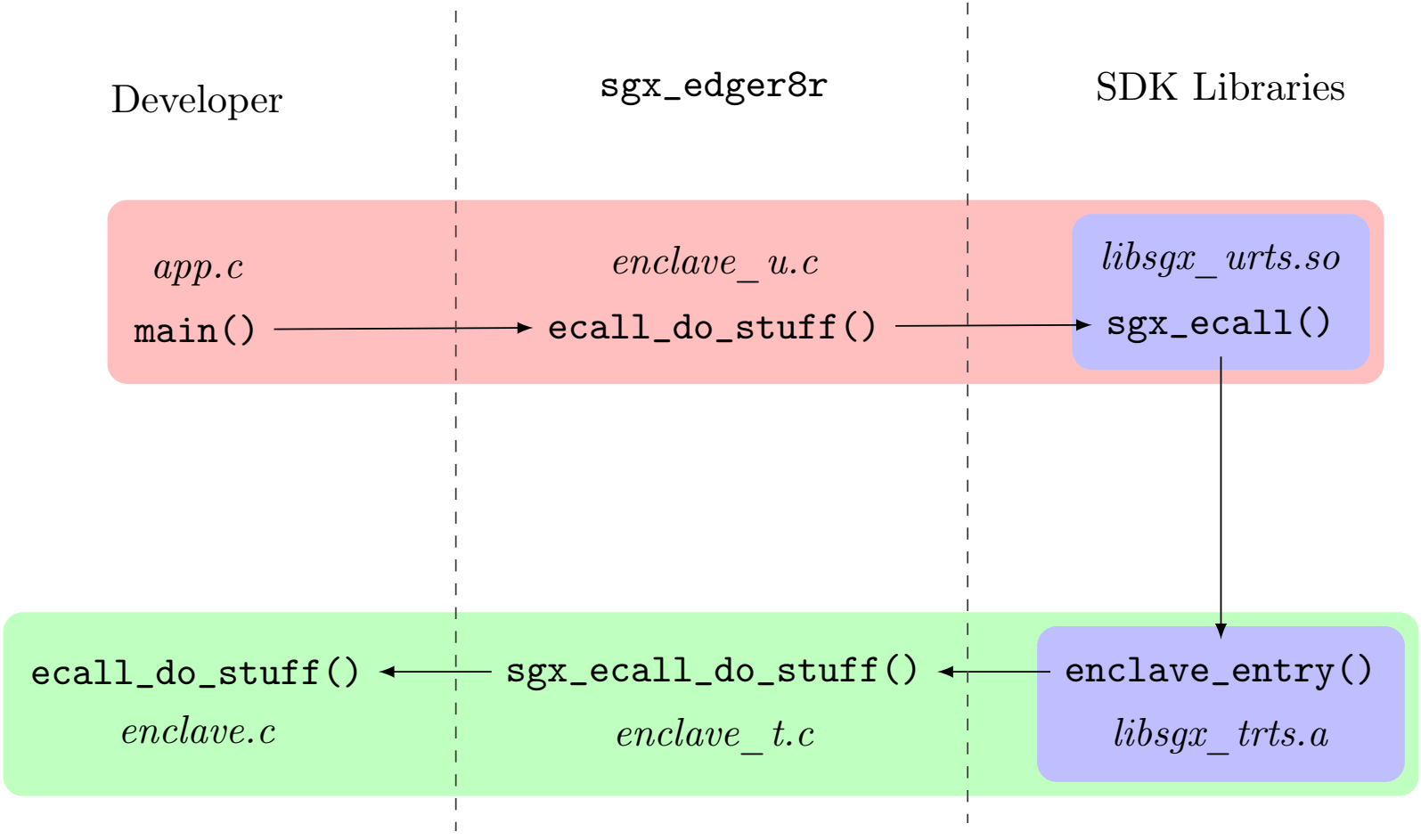
- No high-level performance profiler for SGX exists

Do anti-patterns for enclave programming exist? If so, what are they?

SGX SDK

- SGX SDK to ease development
- Wraps SGX instructions
- Easy way of defining enclave interface
- Call enclave functions like normal functions
  - ECall - from application into enclave
  - OCall - from enclave into application

SGX Ecalls


Developer
sgx_edger8r
SDK Libraries
app.c
main()
enclave_u.c
ecall_do_stuff()
libsgx_urts.so
sgx_ecall()
ecall_do_stuff()
enclave.c
sgx_ecall_do_stuff()
enclave_t.c
enclave_entry()
libsgx_trts.a

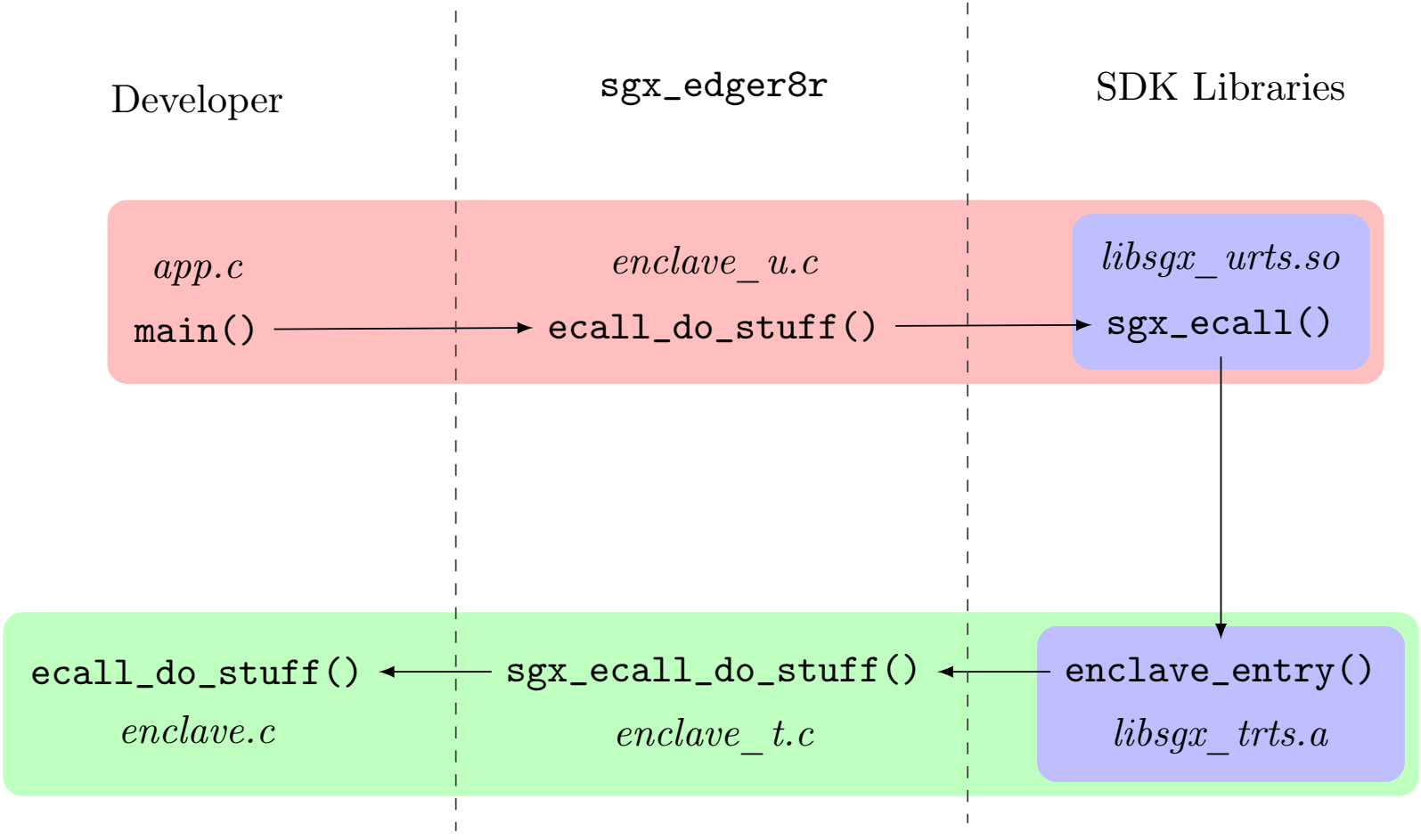
sgx-perf[2]

- Trace enclaves at runtime, analyse trace afterwards
- Identified anti-patterns based on short enclave transitions
- sgx-perf gives recommendations to developers
- Four applications
  - TaLoS (SGX enhanced LibreSSL)
  - SecureKeeper (SGX enhanced ZooKeeper)
  - SQLite
  - Glamdring partitioned LibreSSL

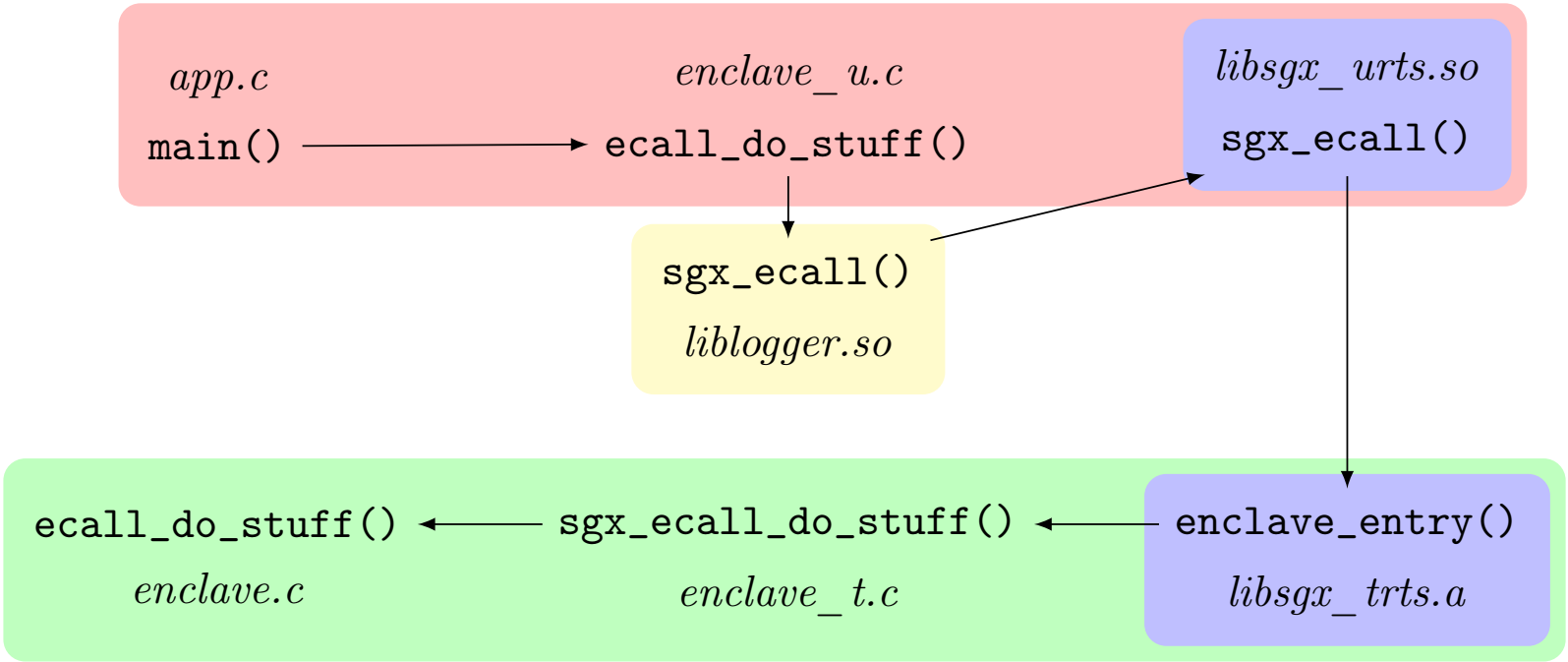
- Well be published end of December

[2] 19th International Middleware Conference (Middleware 2018)

Tracing SGX Applications - ECalls



Tracing SGX Applications - ECalls



Glamdring LibreSSL

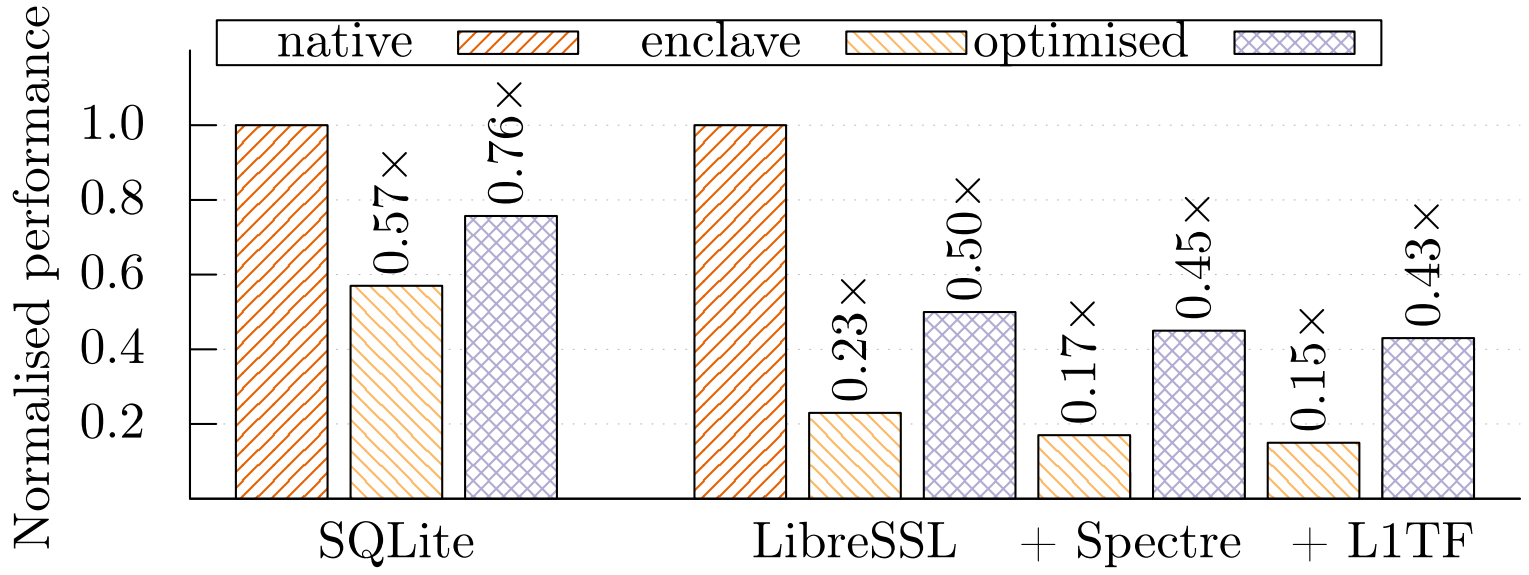
- Automatically partitioned LibreSSL
- 171 ecalls, 3357 ocalls
- 6.6 million ecalls logged, 99.5% to `ecall_bn_sub_part_words`
  - avg. time of ≈3µs (so basically one transition)
- 110 thousand ocalls logged
  - 95% shorter than 10µs

Glamdring LibreSSL

```
void bn_mul_recursive(...) {
    // ...
    switch (c1 * 3 + c2) {
    case -4:
        ecall_bn_sub_part_words(t, a+n, a, tna, tna-n);
        ecall_bn_sub_part_words(t+n, b, b+n, tnb, n-tnb);
        break;
    // ... Repeated three more times
    }
    // ...
}
```

sgx-perf

- 1.33× - 2.16× performance increase by applying recommendations
  - 2.65× Spectre, 2.87× Foreshadow (L1TF)



Paging Revisited

- STANlite addresses paging for DB data, not code
- SQLite is a small database (code)
- Serious DBMS are much bigger than SQLite

  → Code may be subject to paging

- Try to prevent page faults by loading pages before entering enclave
  - E.g., use the variability managers knowledge
  - Or trace page accesses for queries and use
- Pin pages in enclave memory to prevent paging
  - Needs modified SGX driver

Prefetching

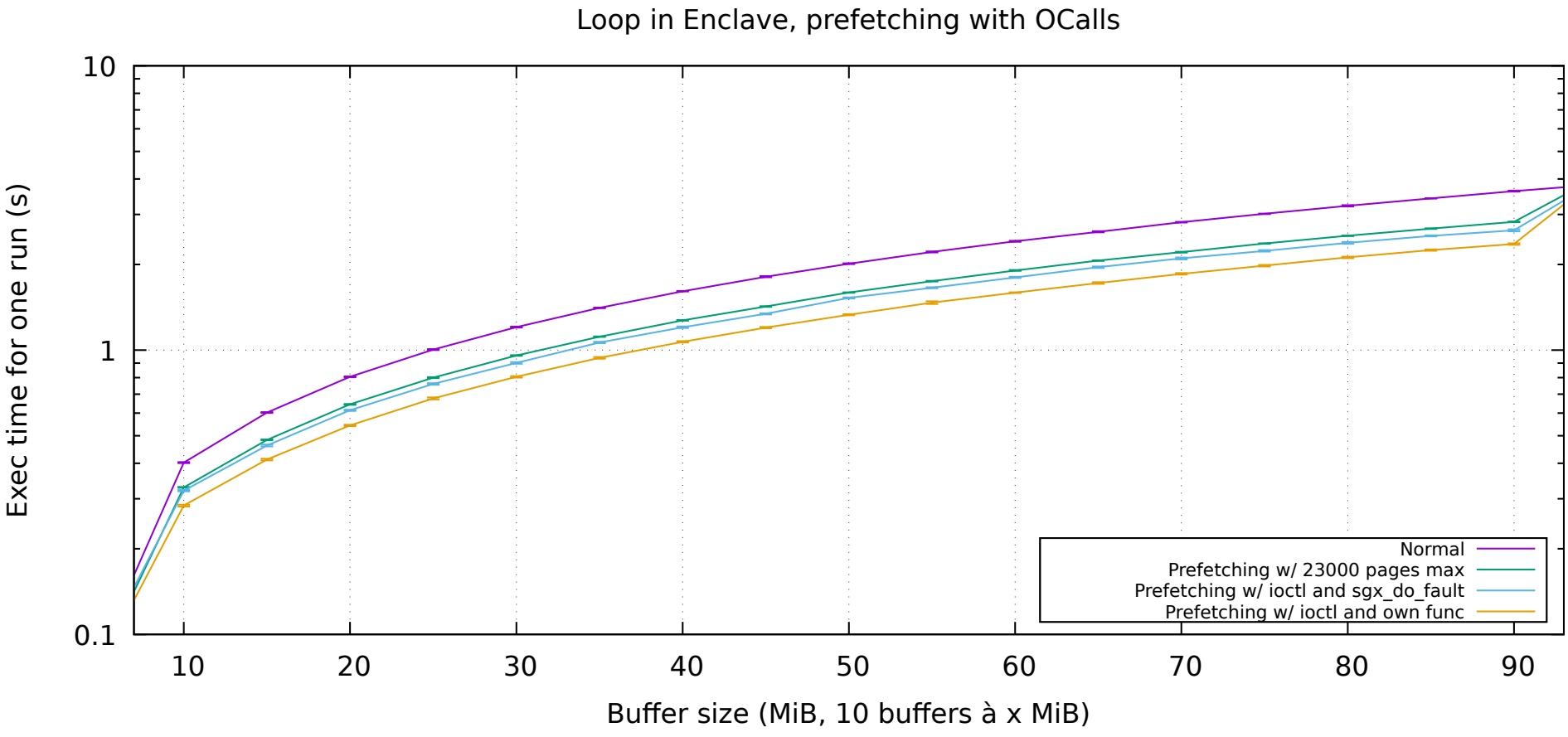
- Trace page accesses and group them into *page sets*
- Prefetch page sets on demand

Prototype:

- Allocate 10 buffers à *x* MB

```
for each buffer
    prefetch(buffer id)
    for each page in buffer
        write to page
    unprefetch(buffer id)
```

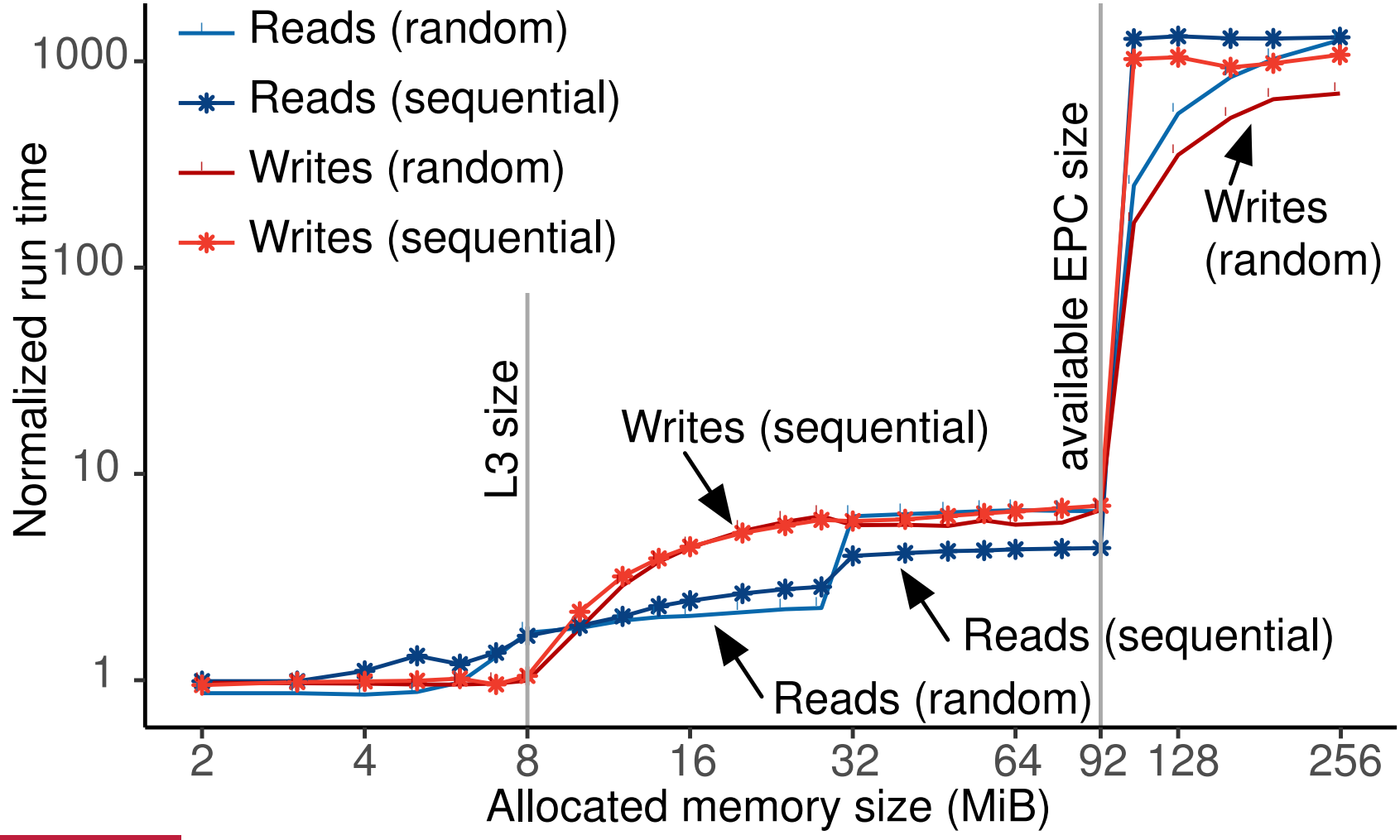
Performance



Prefetching Current State

- 3 modes
  - Touch pages outside, still incurs page fault
  - Tell kernel to prefetch, use page fault function
  - Tell kernel, use own function
- $\approx$ 20-30% faster
- Kernel prefetching is faster
  - But results in more enclave transition than simply touching pages
- Challenge: Identifying page set in a app/DBMS

Backup - SCONE Paging



Short Identical Successive Calls (SISC)

```
void my_untrusted_function()
{
  // ...
  ecall_first(arg1);
  ecall_first(arg2);
  ecall_first(arg3);
  // ...
}
```

```
void my_untrusted_function()
{
  // ...
  arg_type_t args_array[] = {
      arg1, arg2, arg3};
  ecall_first_batch(args);
  // ...
}
```

- Multiple successive calls of the same ecall/ocall
- Short execution time per call (< 10µs)

→ Batching

→ Move caller into or out of enclave

Short Different Successice Calls (SDSC)

```
void my_untrusted_function()
{
  // ...
  ecall_first(arg1);
  ecall_second(arg2);
  ecall_third(arg3);
  // ...
}
```

```
void my_untrusted_function()
{
  // ...
  ecall_merged(arg1, arg2,
      arg3);
  // ...
}
```

- Multiple successive calls of different ecalls/ocalls
- Short execution time per call (< 10µs)

→ Merging

→ Move caller into or out of enclave

Short Nested Calls (SNC)

```
void my_ecall()
{
    ocall_first();
    // ...
    ocall_second();
}
```

```
void my_untrusted_function()
{
    first();
    my_ecall();
    second();
}
```

- Calls at the start or end of other calls
- Short execution time per call (< 10µs)

→ Reorder

→ Duplicate ocall inside the enclave

Short Synchronisation Calls (SSC)

```
sgx_thread_mutex_t mutex;

void my_multithreaded_ecall()
{
    // ...
    sgx_thread_mutex_lock(&mutex);
    // ...
    sgx_thread_mutex_unlock(&mutex);
    // ...
}
```

- Leave enclave if lock is taken
- Short wakeup ocalls (< 10µs)

→ In-enclave spinlocks

→ Hybrid locks

Backup - Enclave transitions prefetching