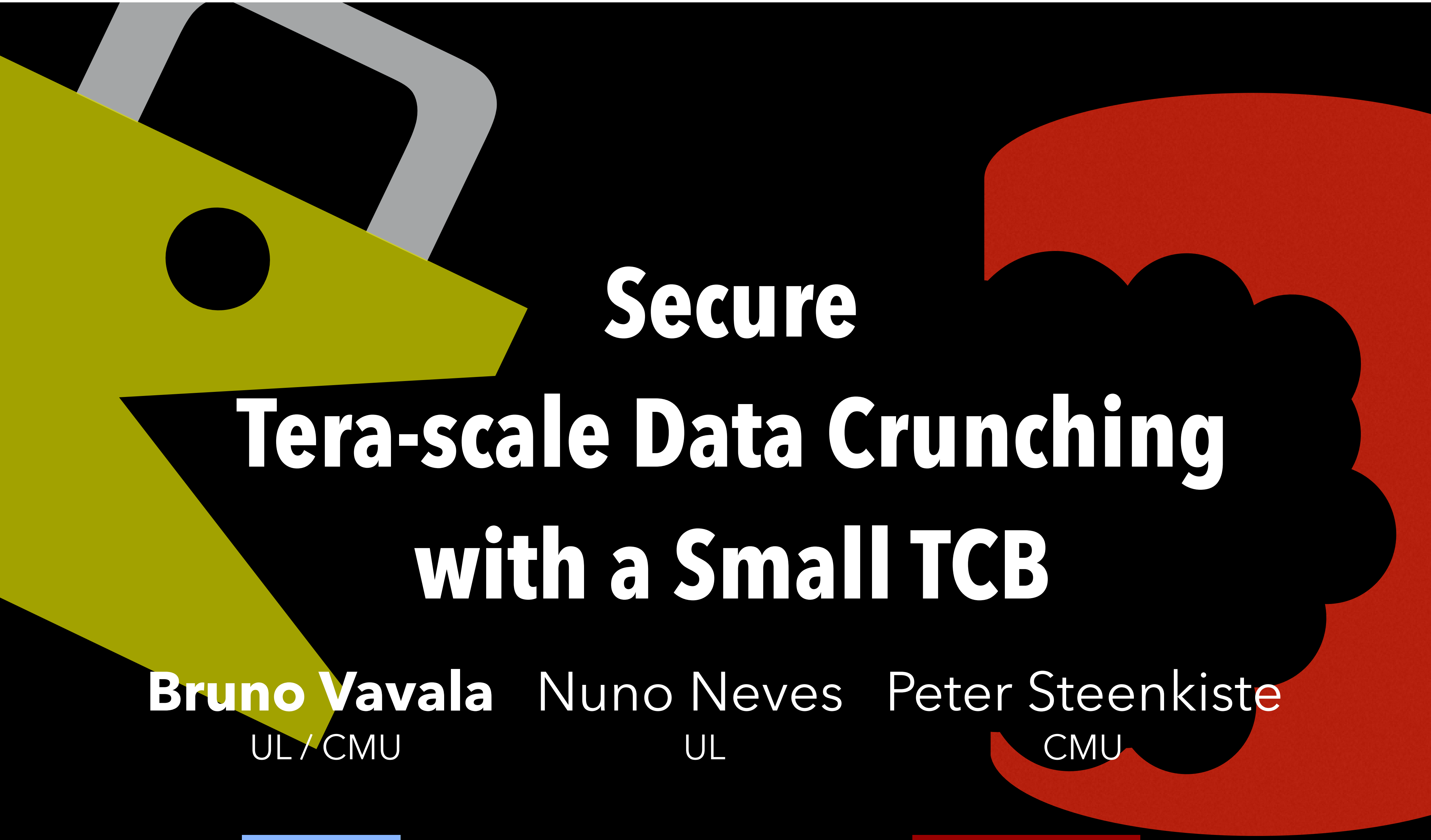


Secure Tera-scale Data Crunching with a Small TCB

Bruno Vavala UL / CMU

Nuno Neves UL

Peter Steenkiste CMU

Ciências ULisboa

Carnegie Mellon University

47th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'17)

Goal

delivering **security guarantees** for **large-scale data processing** on untrusted hosts with a **small TCB**

security guarantees

trusted HW based

data integrity

1 TB

large-scale data processing

small
code

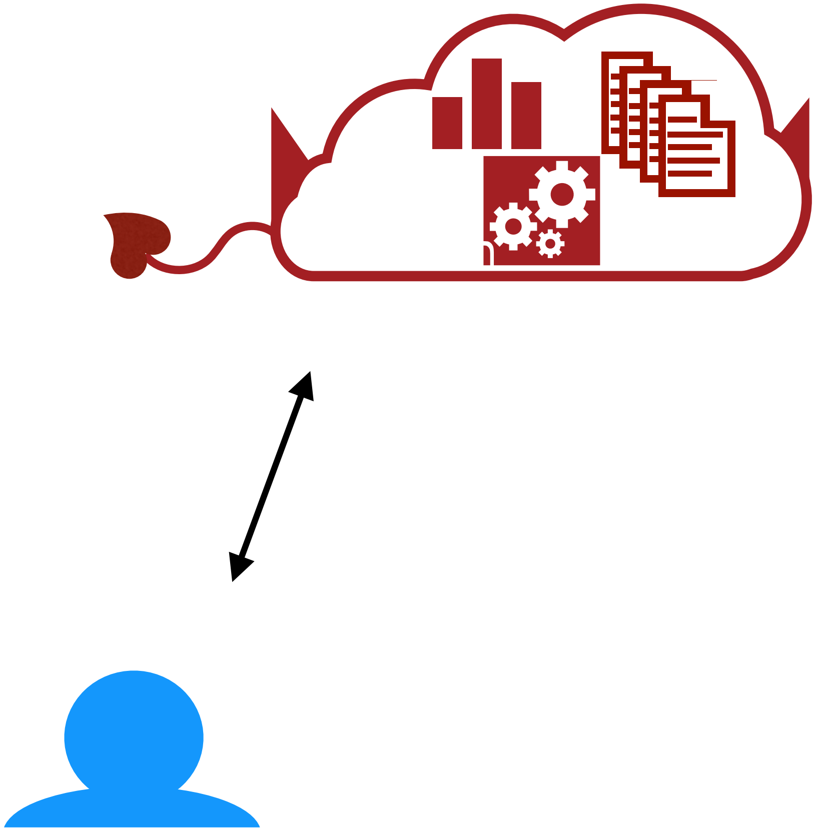
small
interface

small TCB

No HW
devices

Some use cases

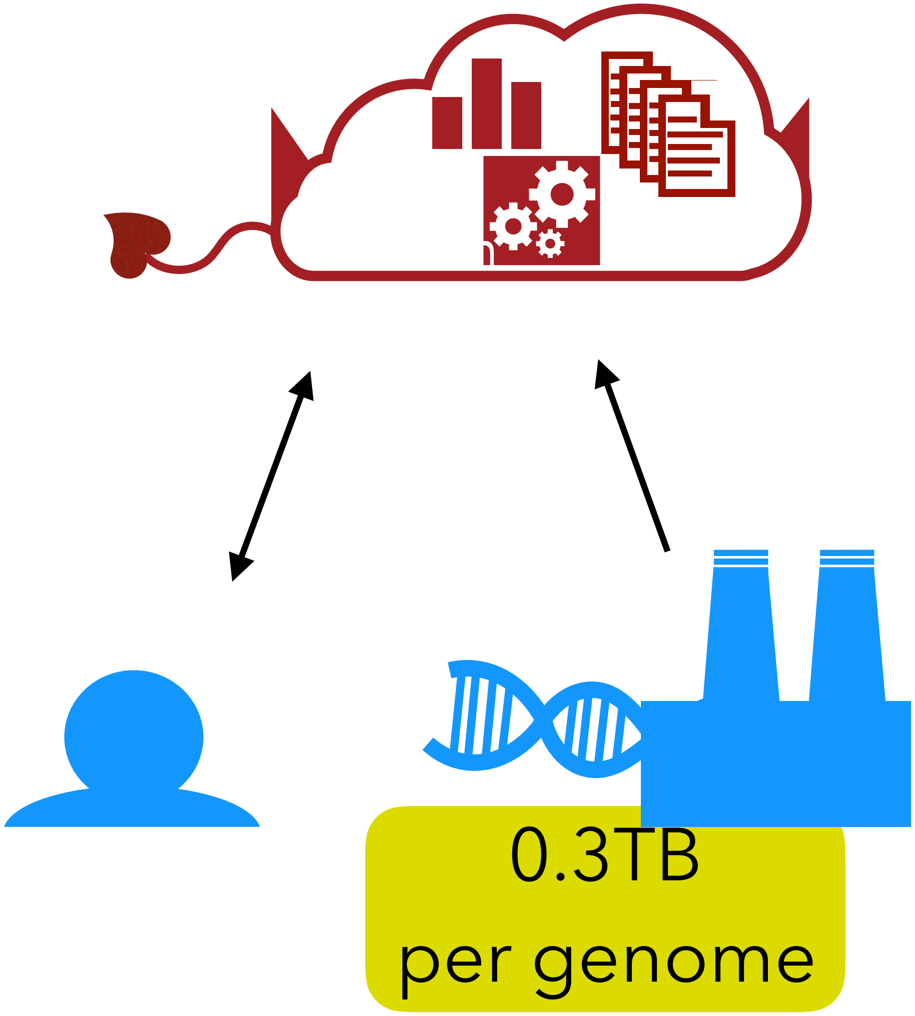
public cloud service provider



Some use cases

public cloud service provider

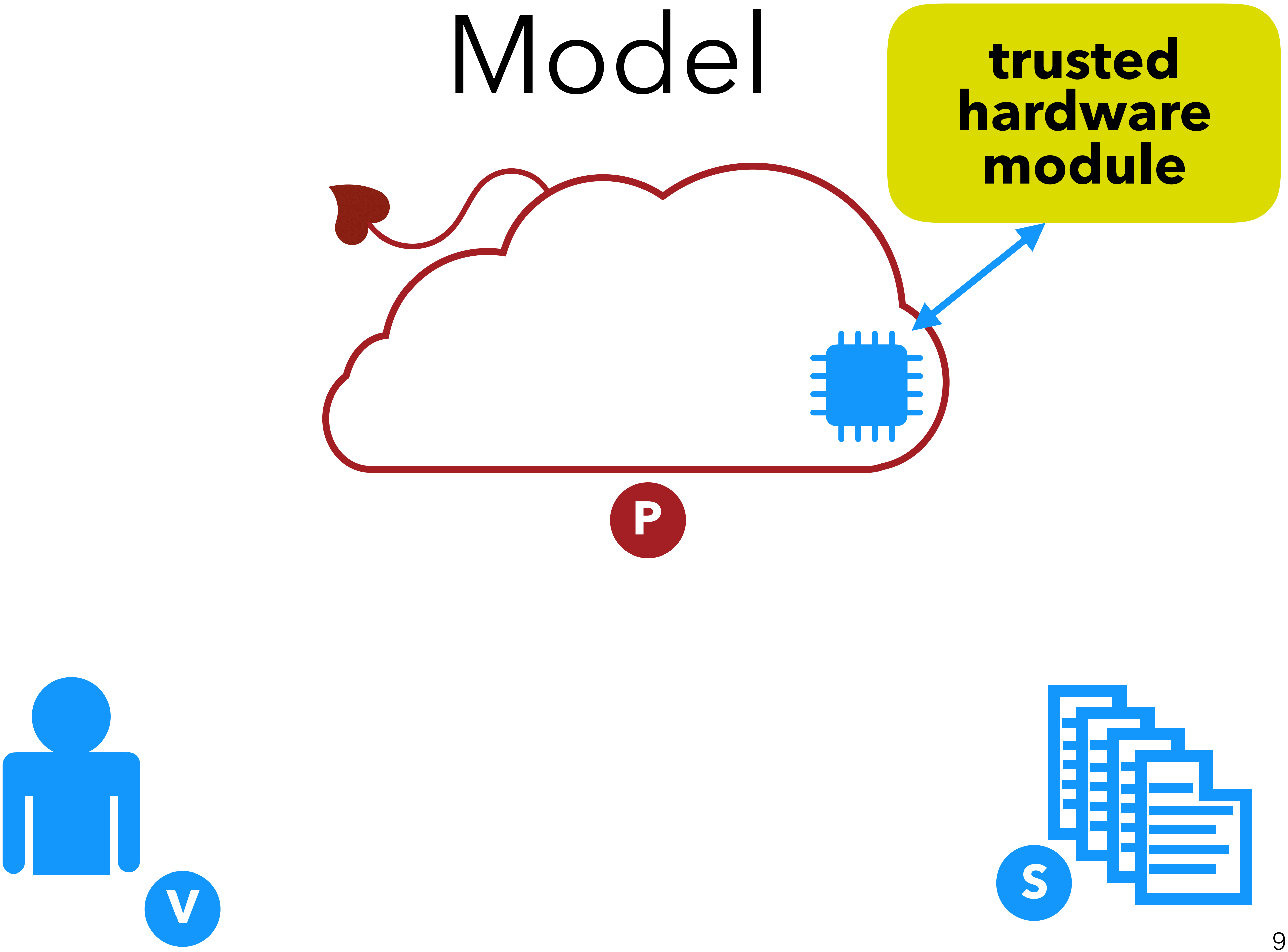
computational genomics



0.3TB per genome

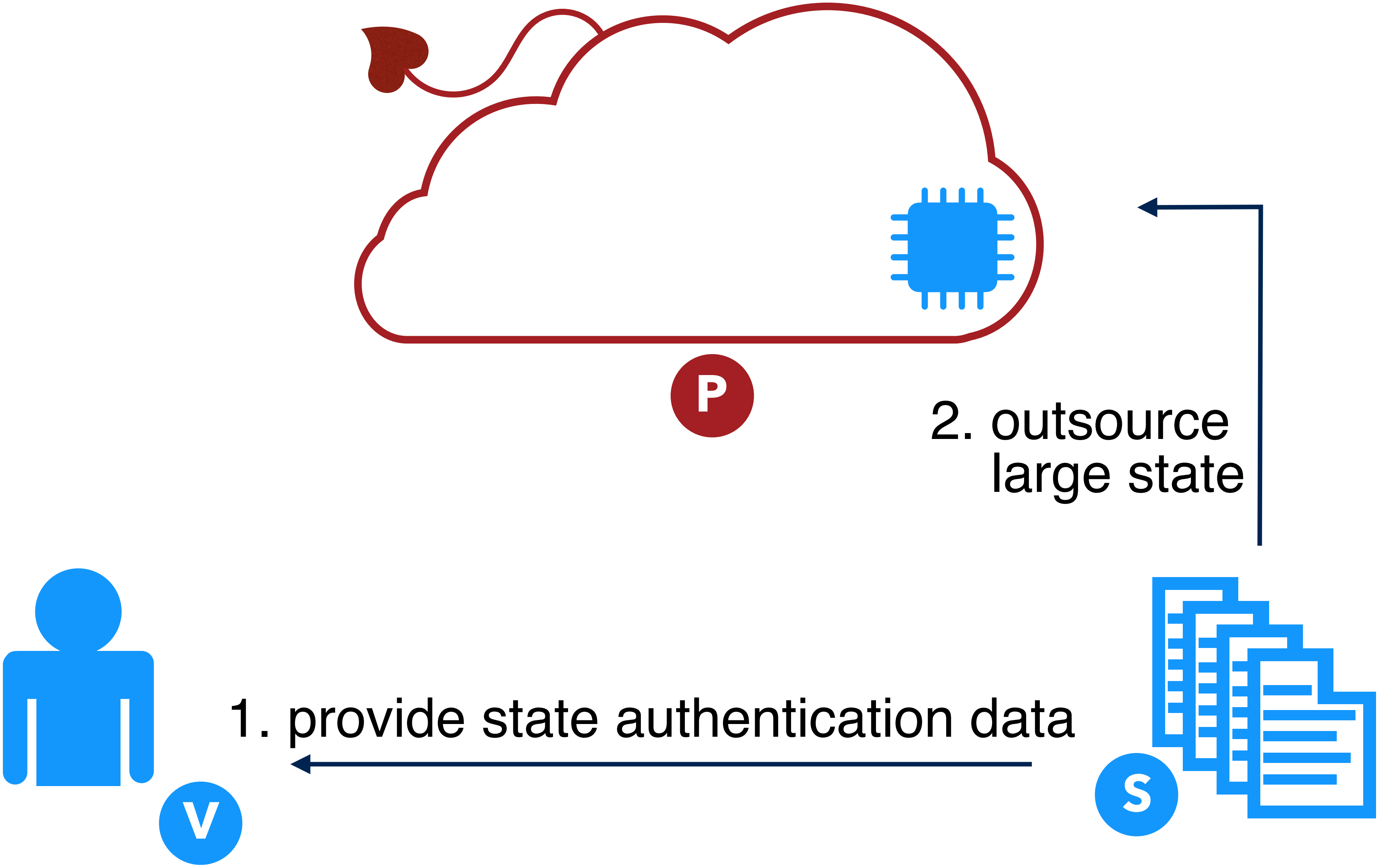
…more generally…

Model



trusted hardware module

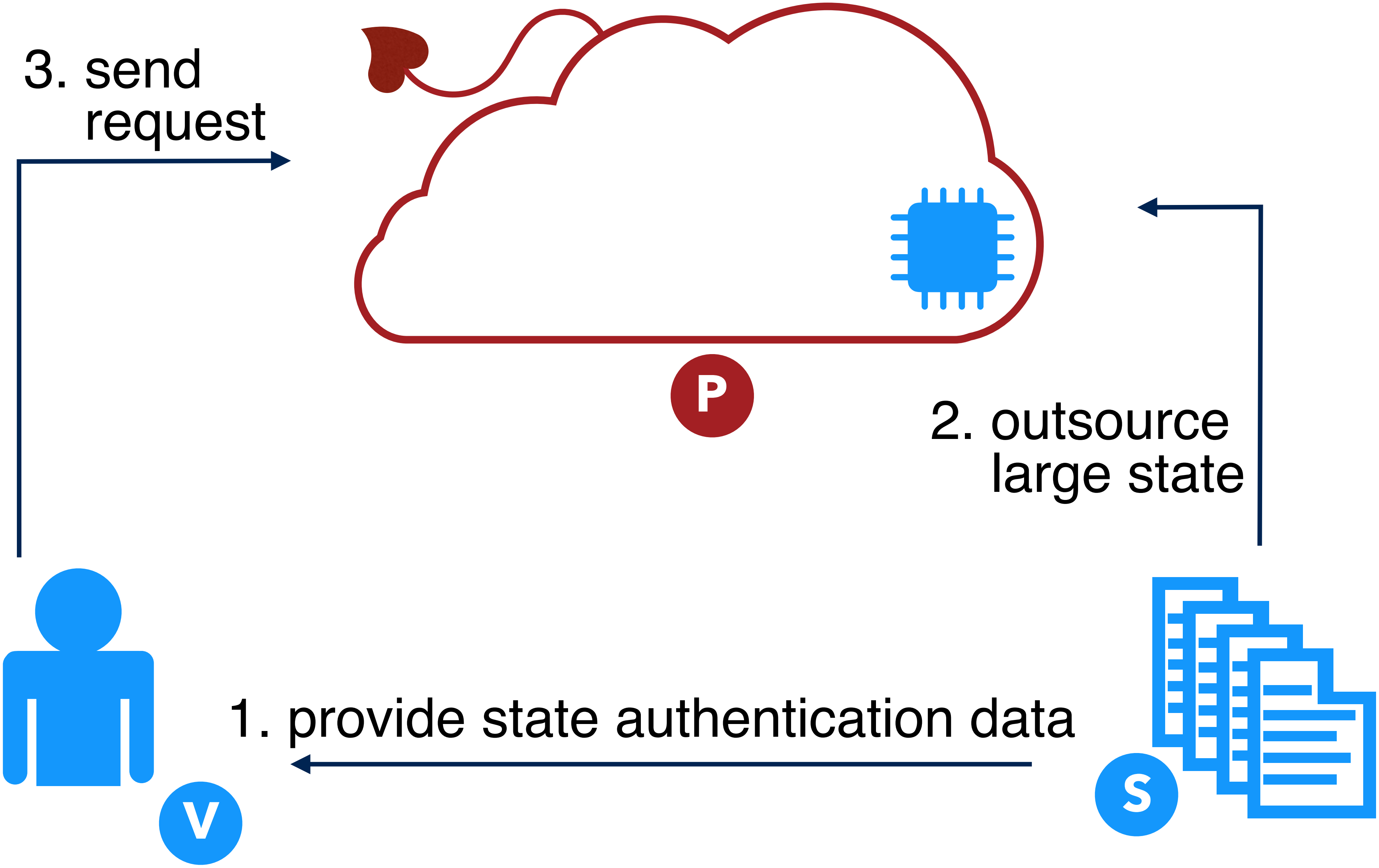
Model



2. outsource large state

1. provide state authentication data

Model

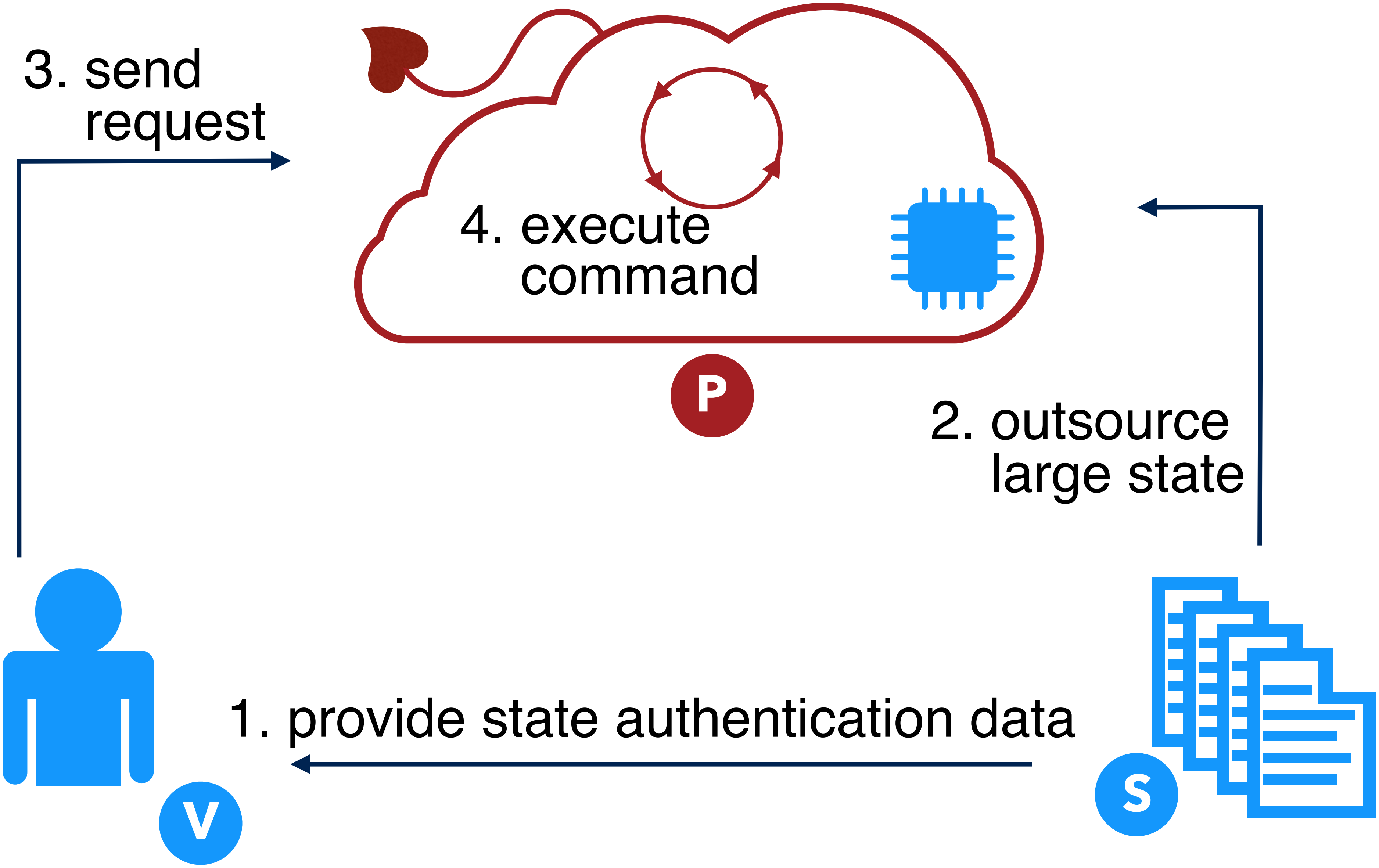


3. send request

2. outsource large state

1. provide state authentication data

Model



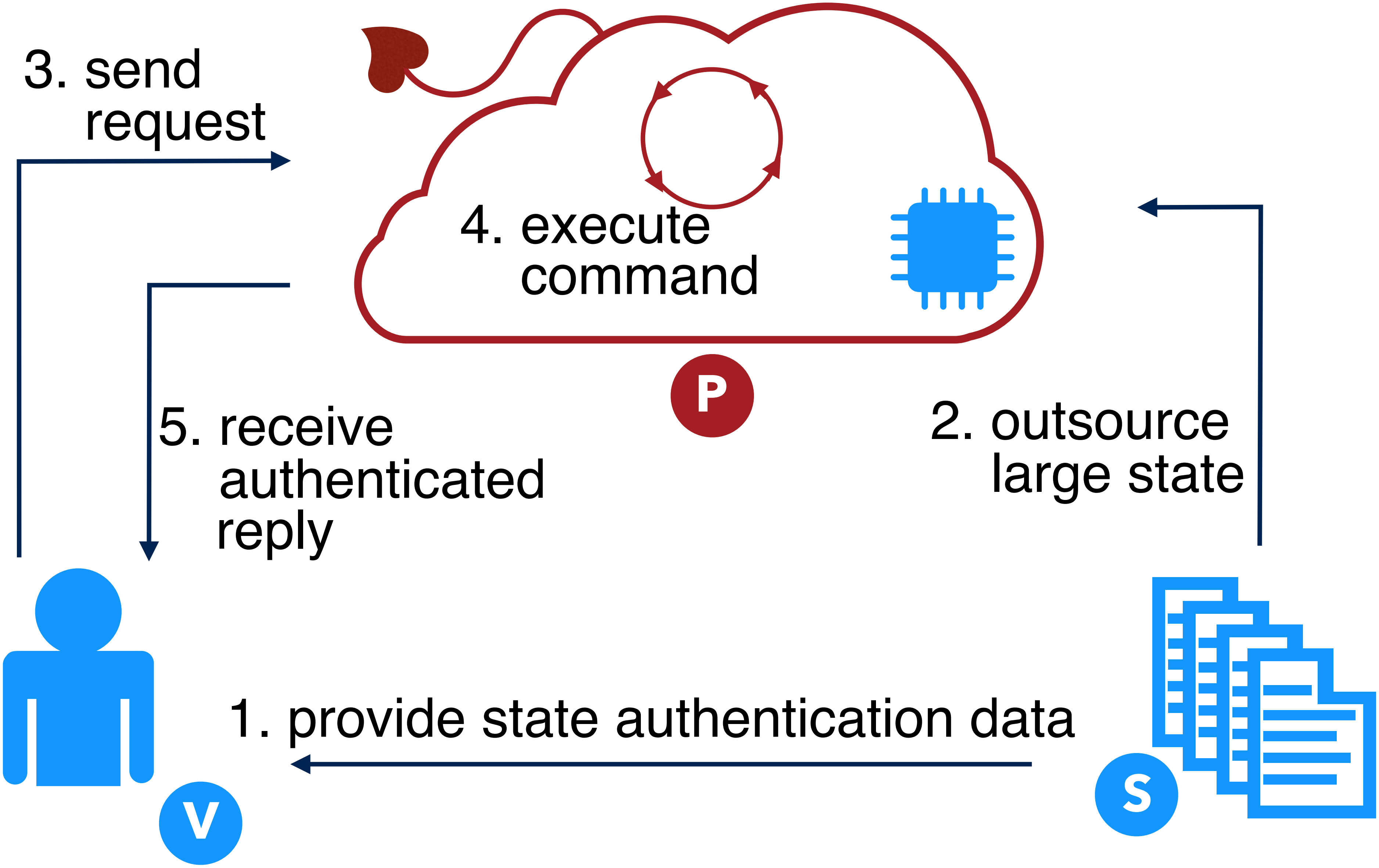
3. send request

4. execute command

2. outsource large state

1. provide state authentication data

Model



3. send request

4. execute command

P

5. receive authenticated reply

2. outsource large state

1. provide state authentication data

V

S

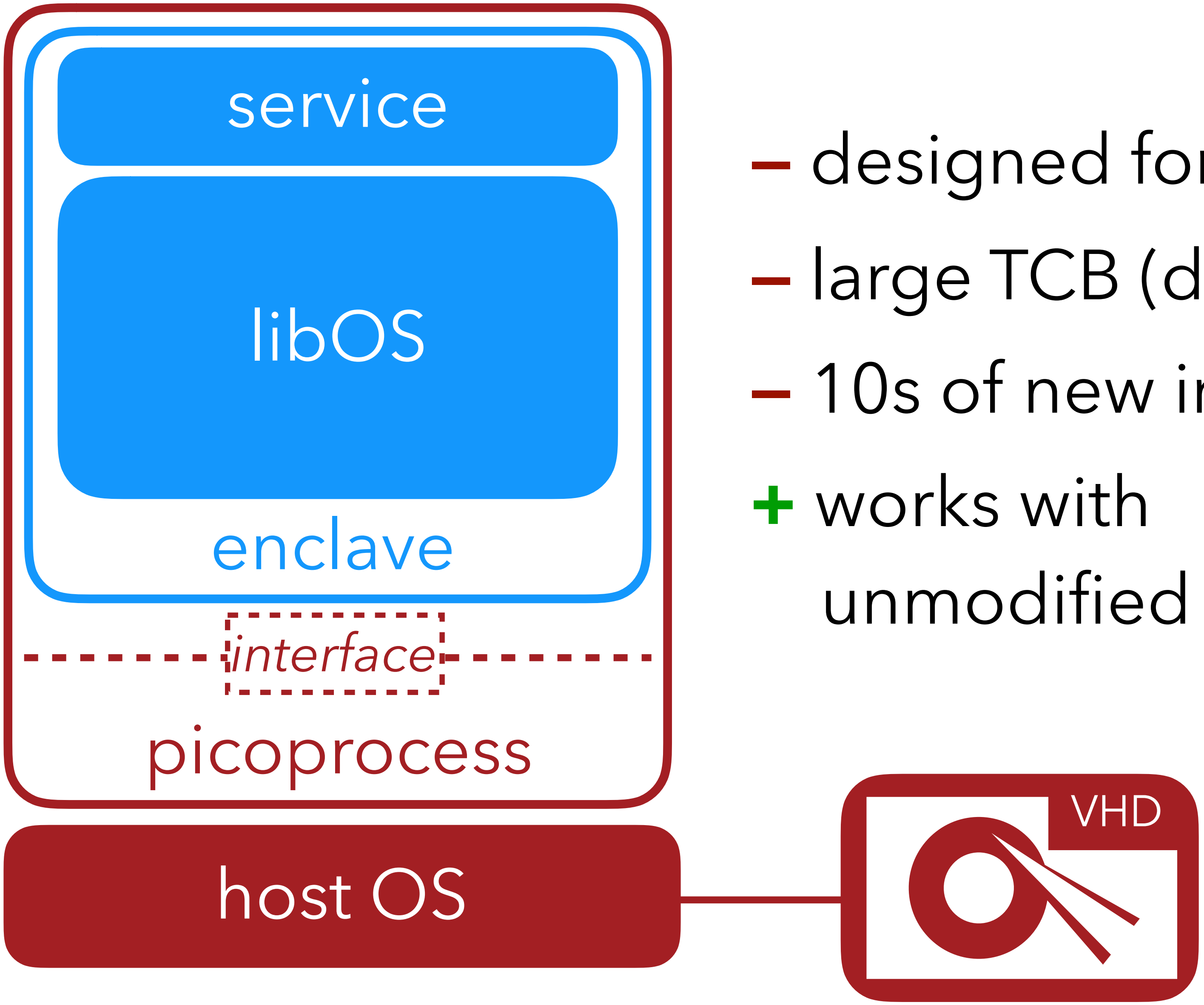
Outline

- **Goal**
- Previous Work
- Our solution: key ideas and overview
- Evaluation

Outline

- **Previous Work**

Haven
(OSDI'14)



– designed for Intel SGX

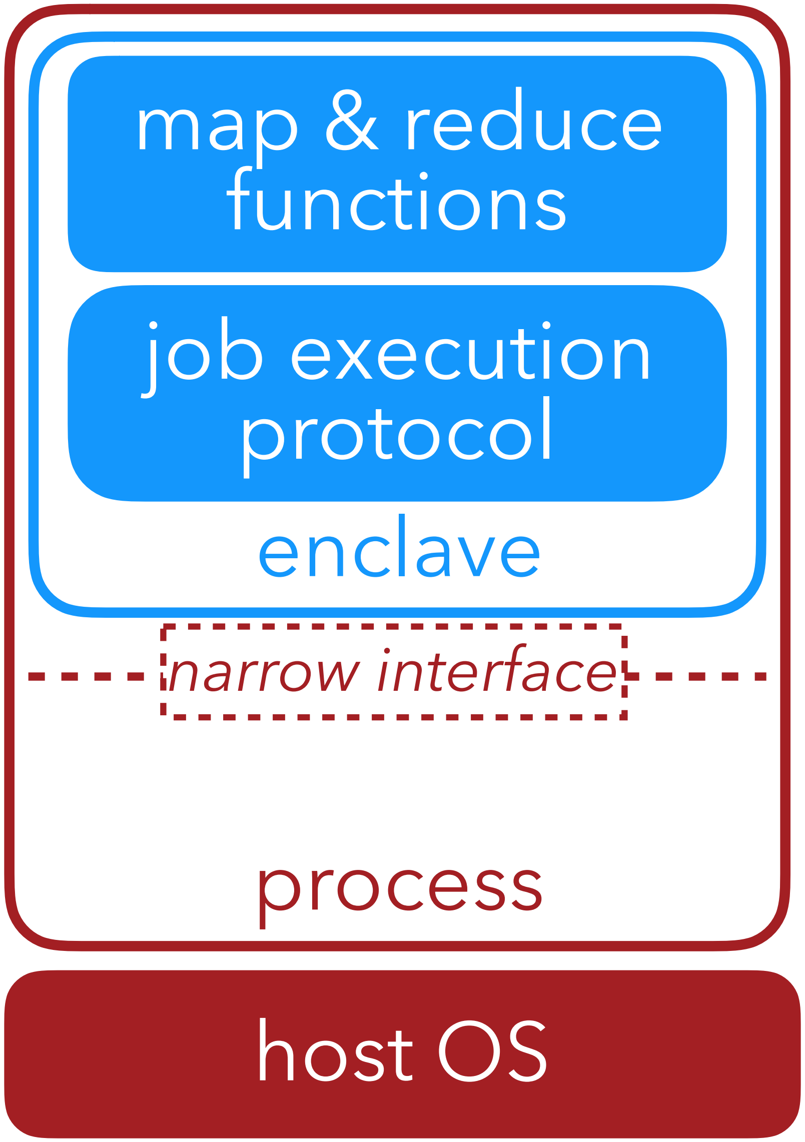
– large TCB (due to libOS)

– 10s of new interface calls

+ works with unmodified applications

VC^3

(IEEE S&P'15)



– designed for Intel SGX

– specific for Hadoop

+ small TCB

+ data confidentiality

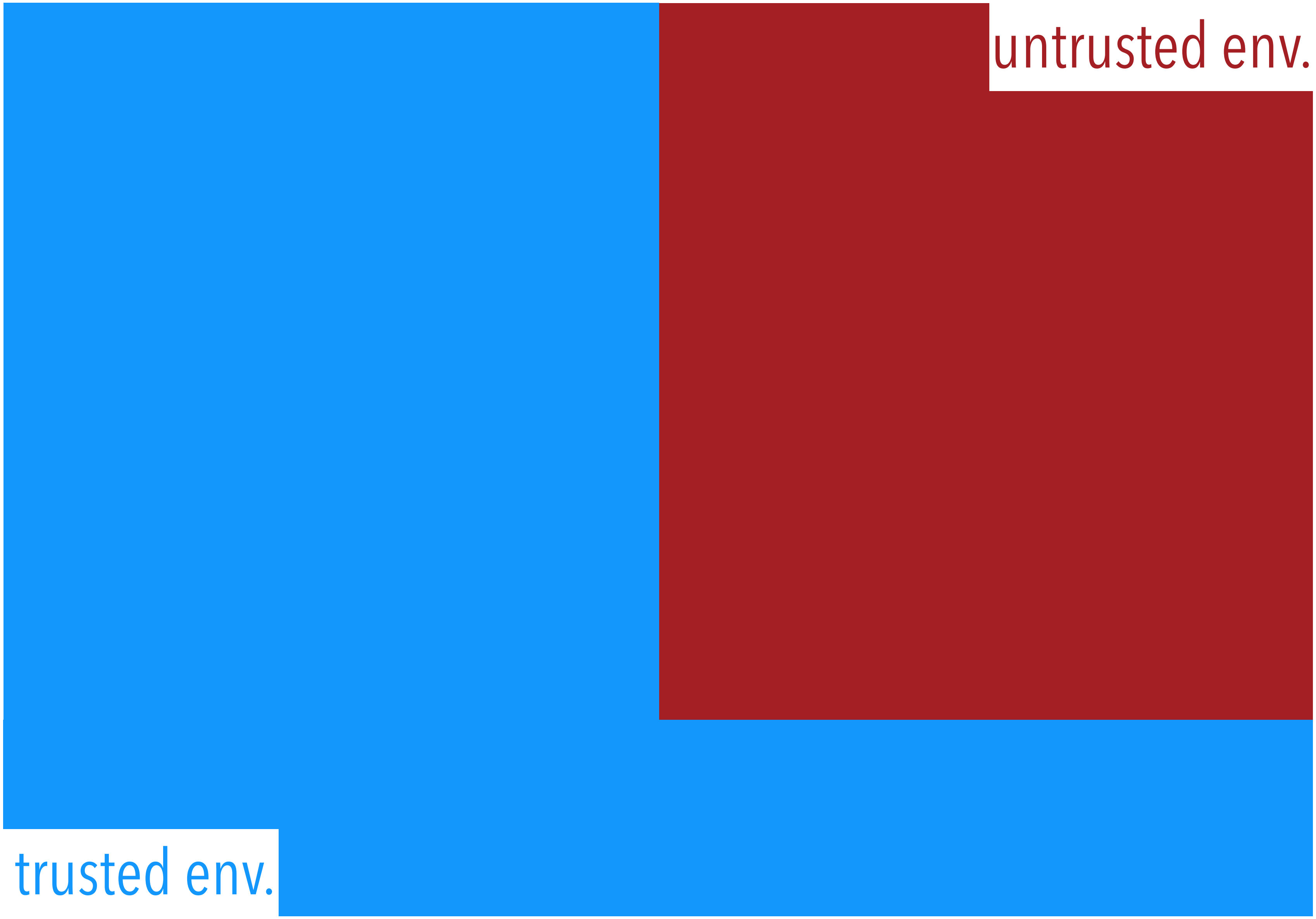
+ can run unmodified Hadoop applications

A Niche in the State of the Art

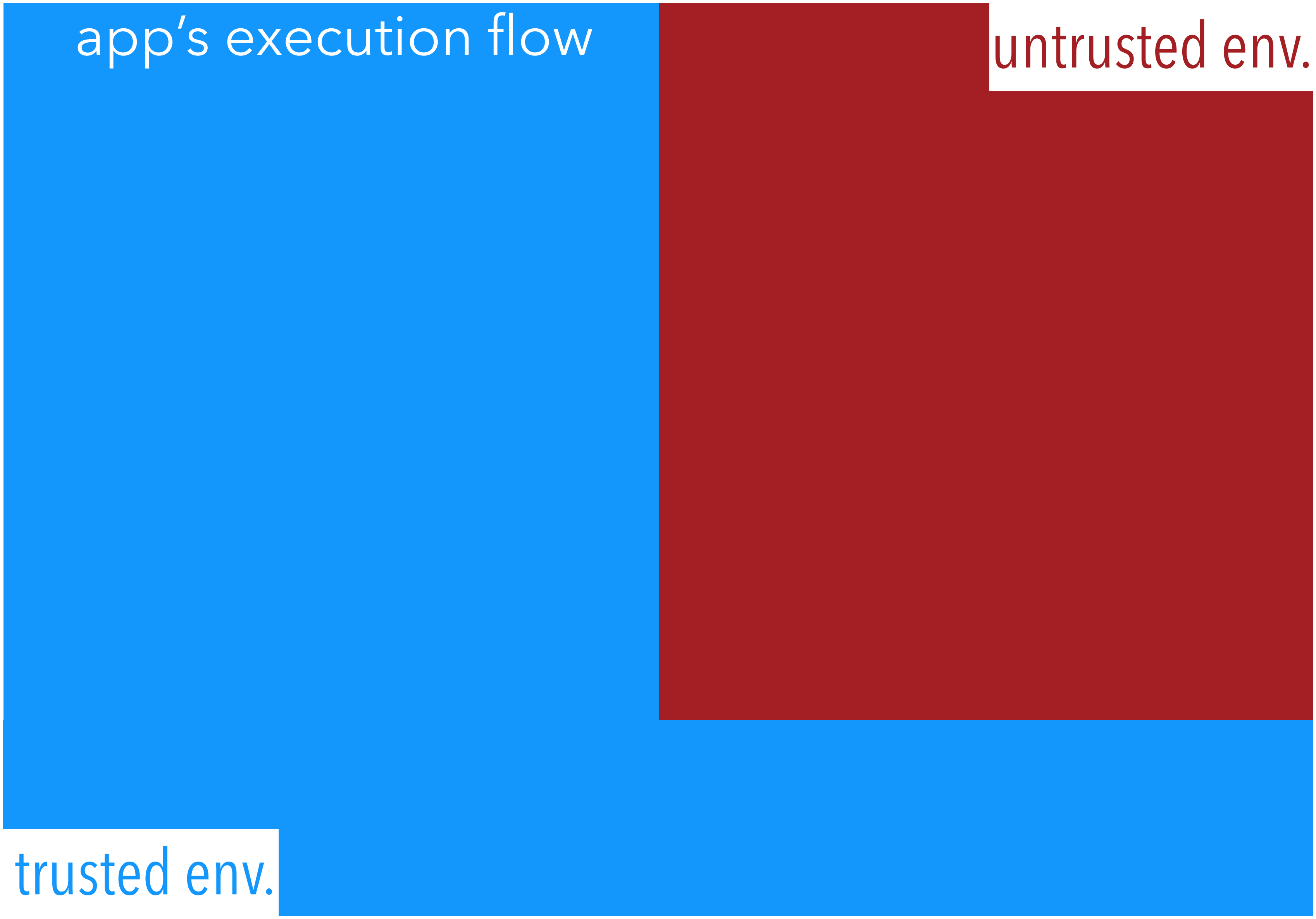
	small TCB	Large State	Interface calls	App Specific	Trusted Computing arch.
Haven (OSDI'14)	No	Yes	tens	No	SGX
VC3 (S&P'15)	Yes	Yes MapReduce workloads	R,W	Yes	SGX
XMHF-TrustVisor (S&P'13,'10)	Yes	No	none (but Minibox has tens)	No	TPM / TXT
LaSt^{GT}	Yes	Yes	zero!	No	TV&SGX

Outline

- **Our solution: key ideas and overview**



Scenario: two execution environments

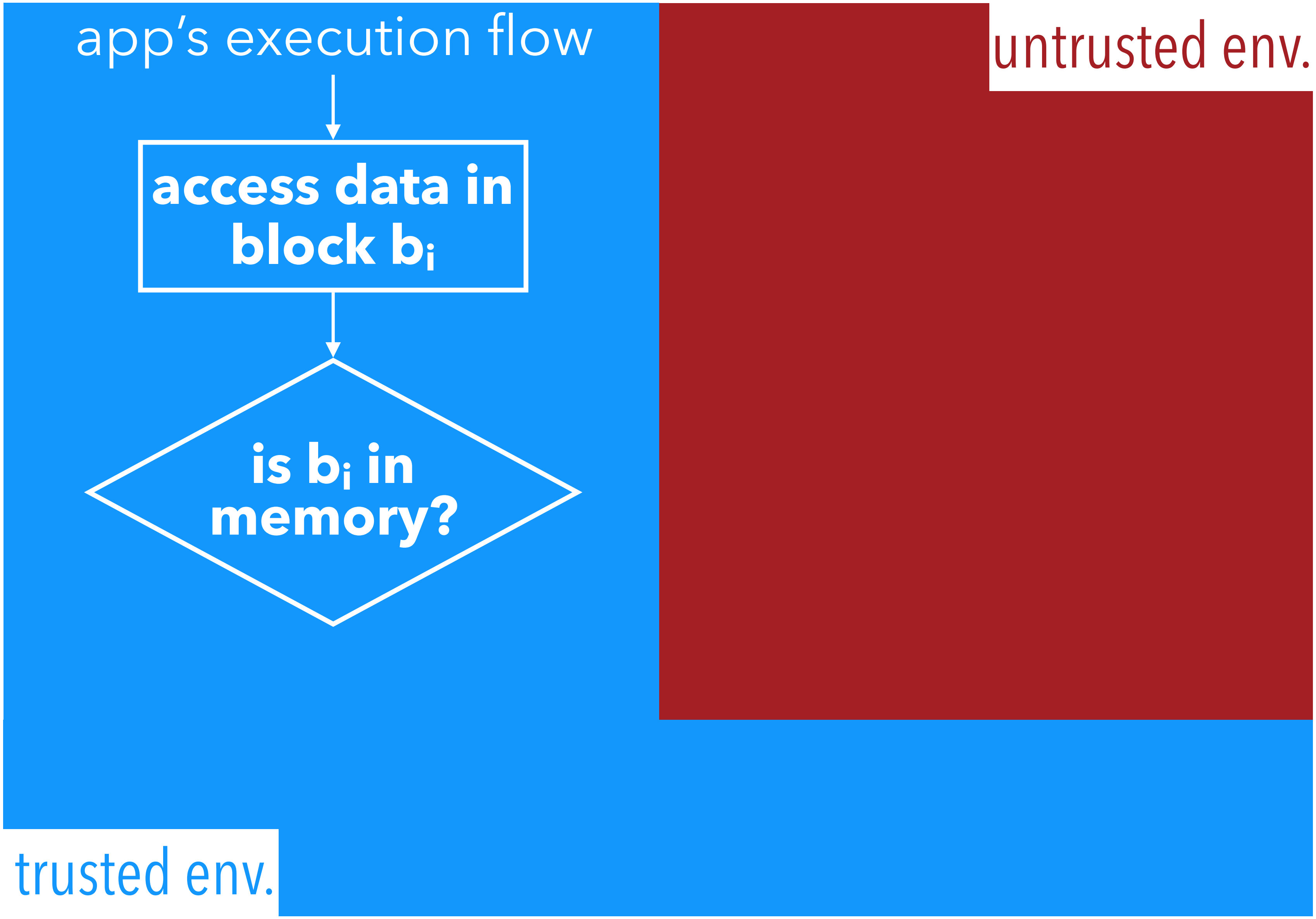


app’s execution flow

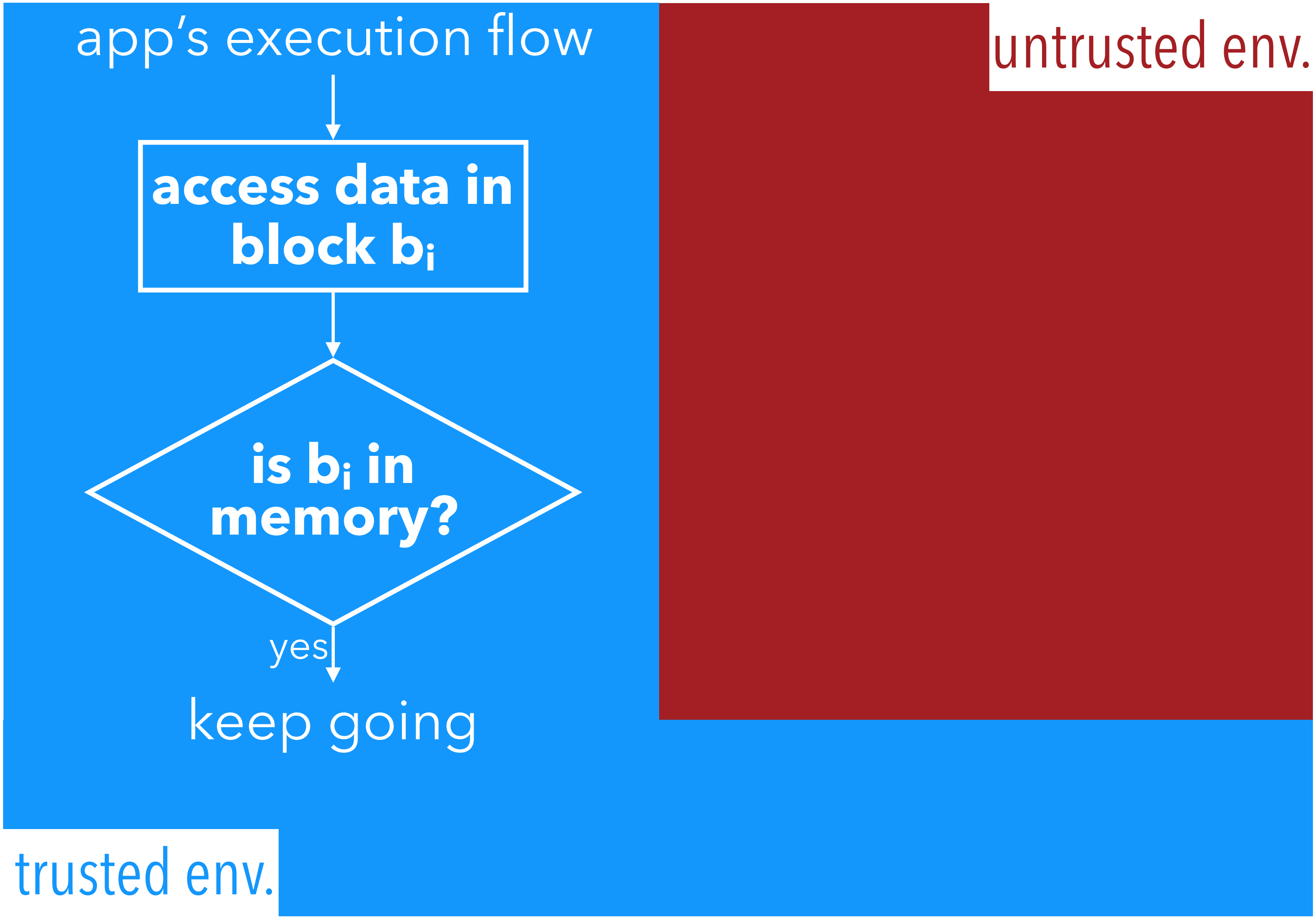
untrusted env.

trusted env.

the service code is running



the service code accesses data in memory



untrusted env.

app's execution flow

access data in block b_i

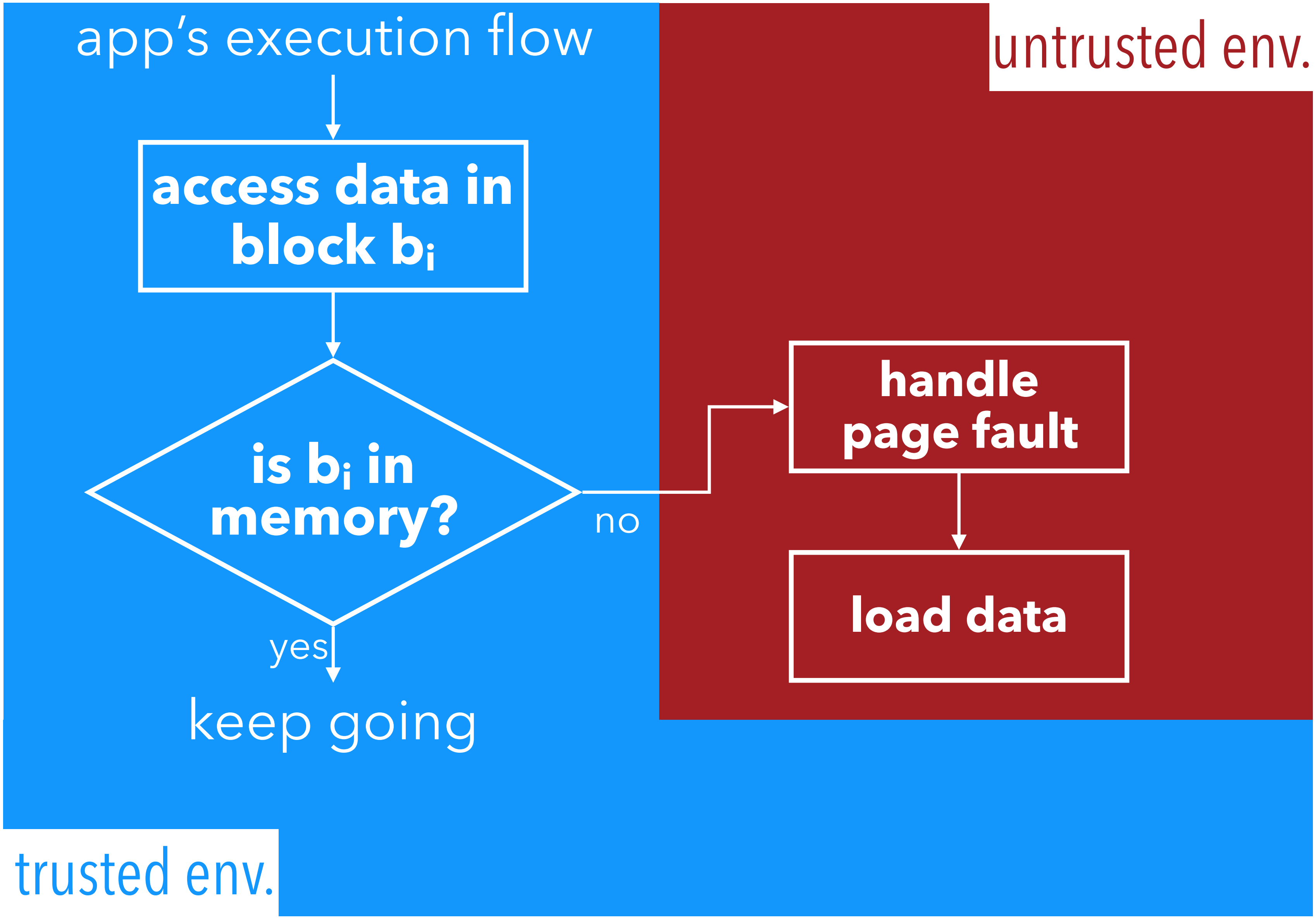
is b_i in memory?

yes

keep going

trusted env.

when data is available, there are no interruptions



app's execution flow

access data in block b_i

is b_i in memory?

yes

keep going

no

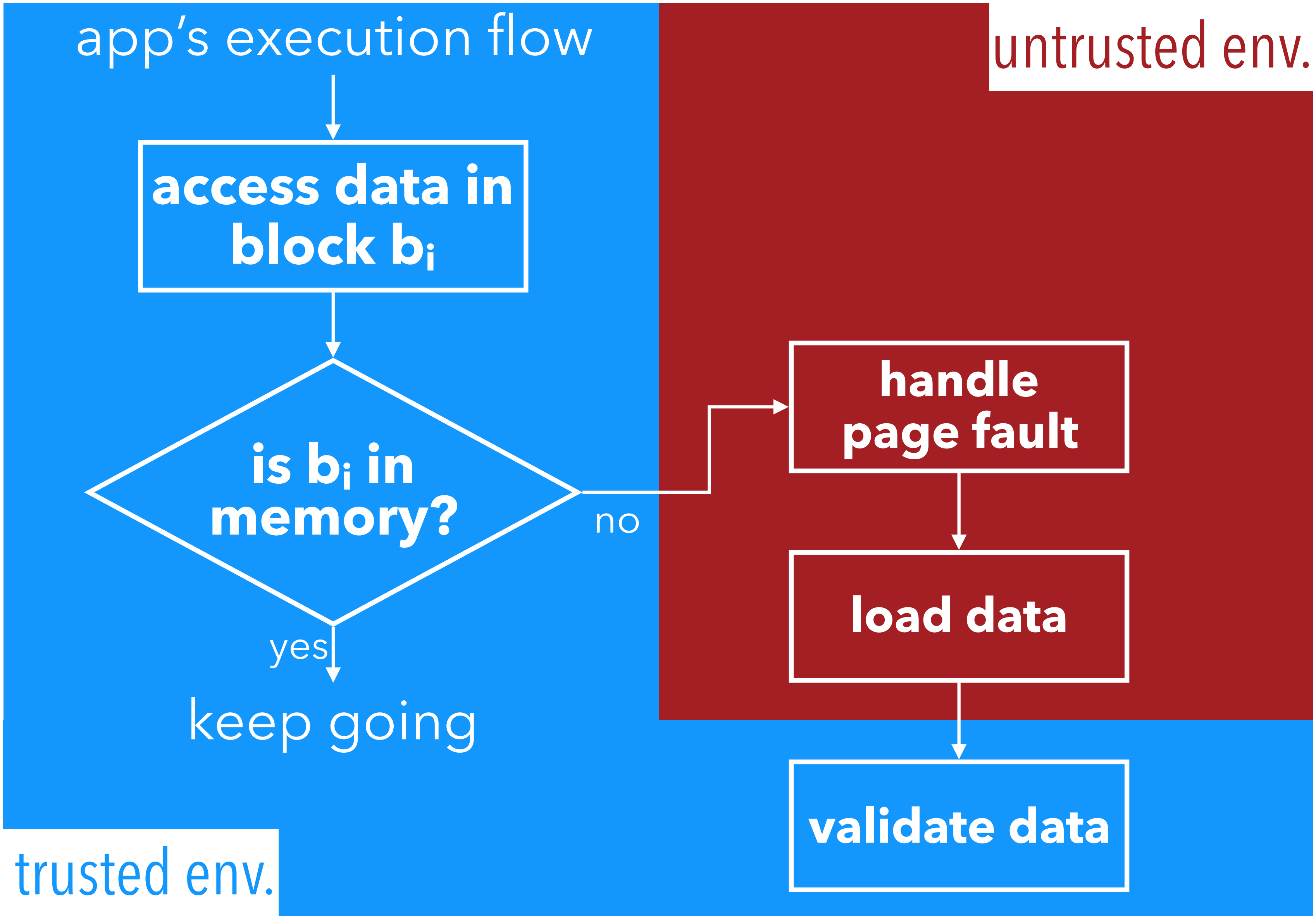
handle page fault

load data

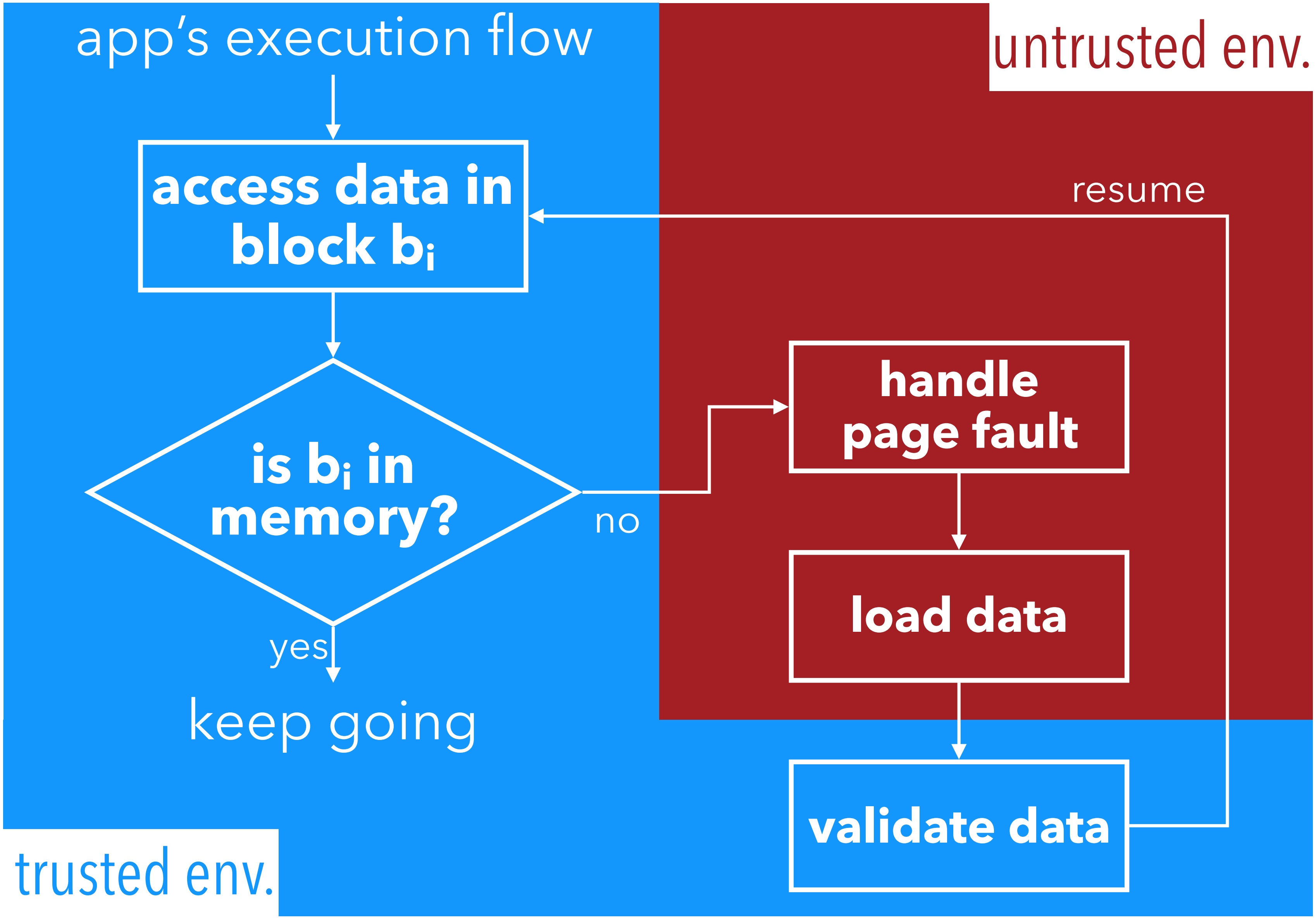
untrusted env.

trusted env.

otherwise, the service is interrupted and data memory pages are loaded



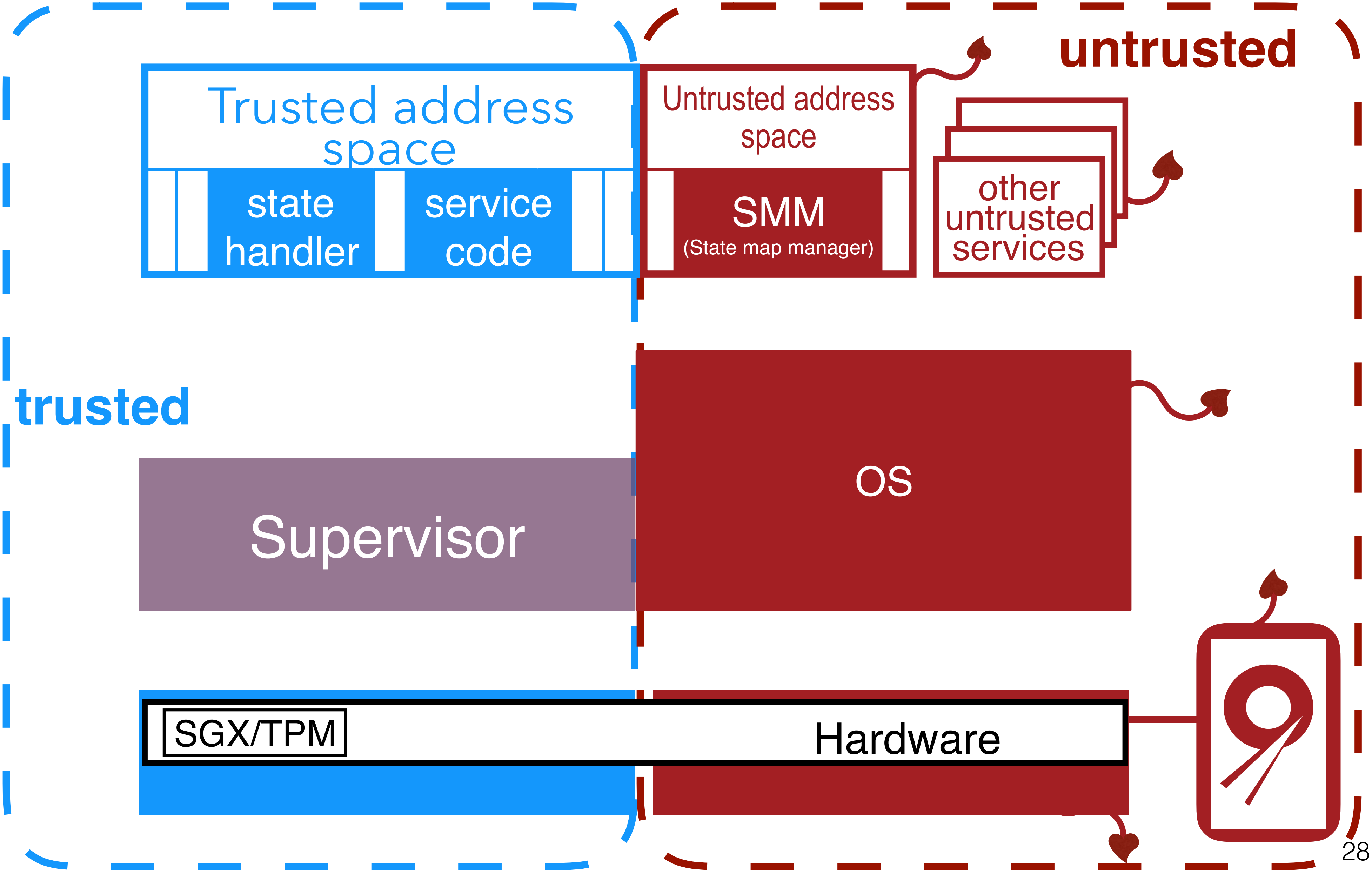
data is validated inside trusted environment, independently from service execution



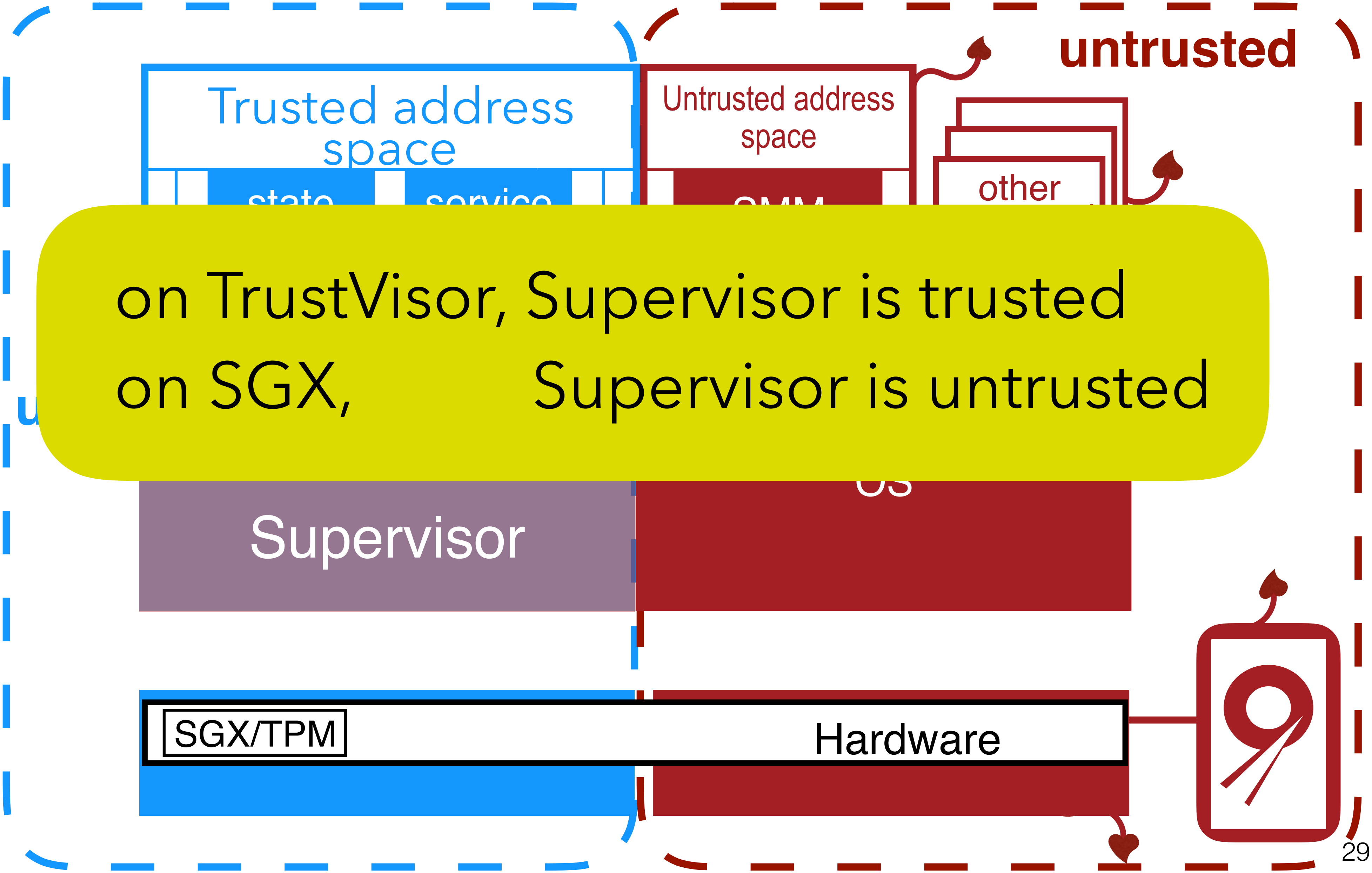
service is resumed and
only if data is valid, service can make progress

…in practice…

Architecture



Architecture



on TrustVisor, Supervisor is trusted

on SGX, Supervisor is untrusted

LaSt^{GT} in ~~5~~ 4 steps

- Offline data protection at the source
- State registration
- Data processing
- Lazy loading from memory & disk
- ~~Execution verification~~

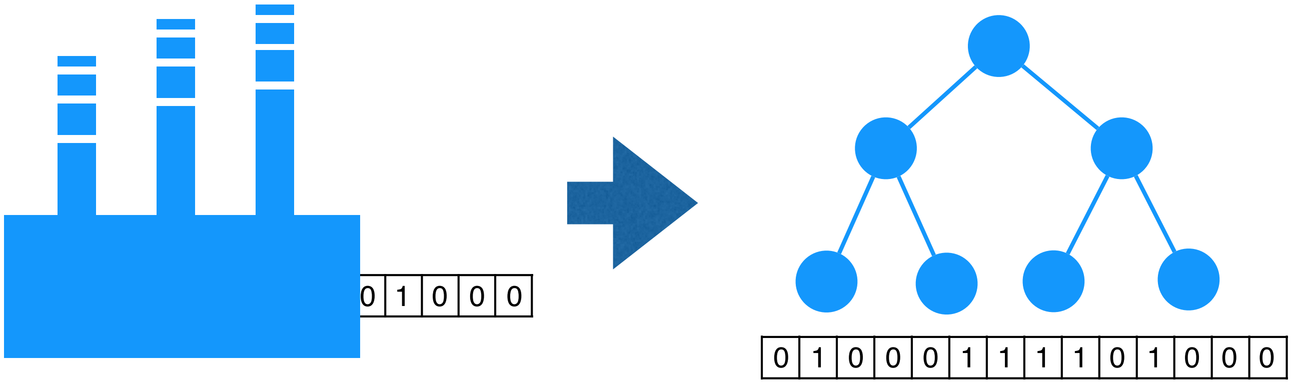
- Offline data protection at the source

1

Data protection

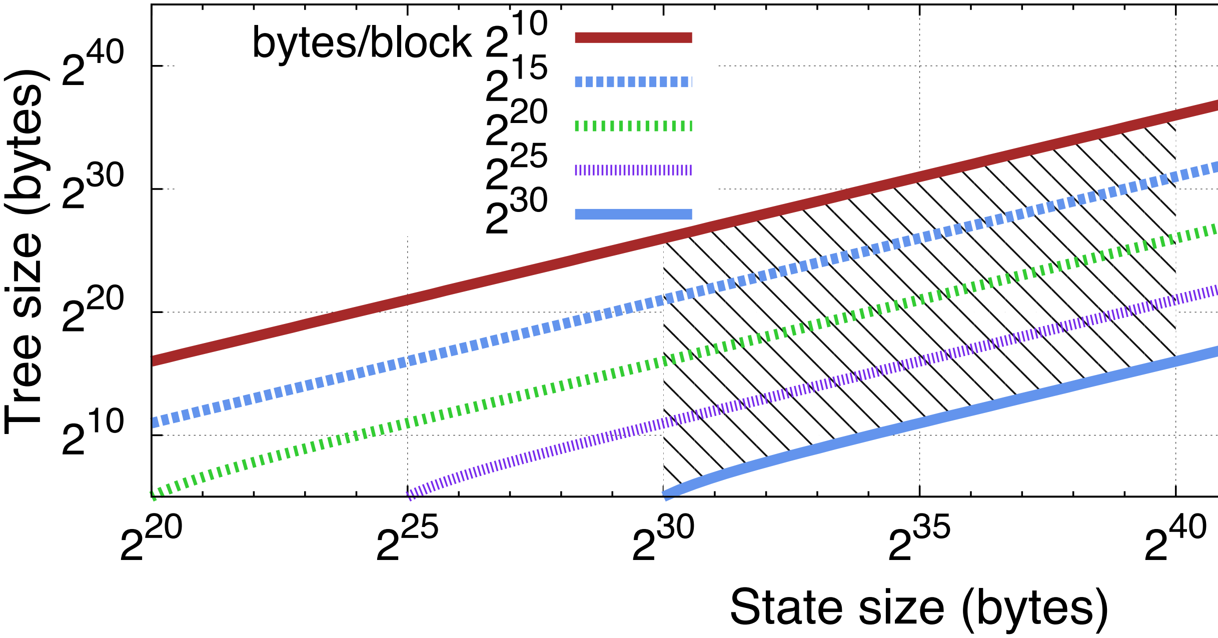
Hierarchical

- Incremental as data is created

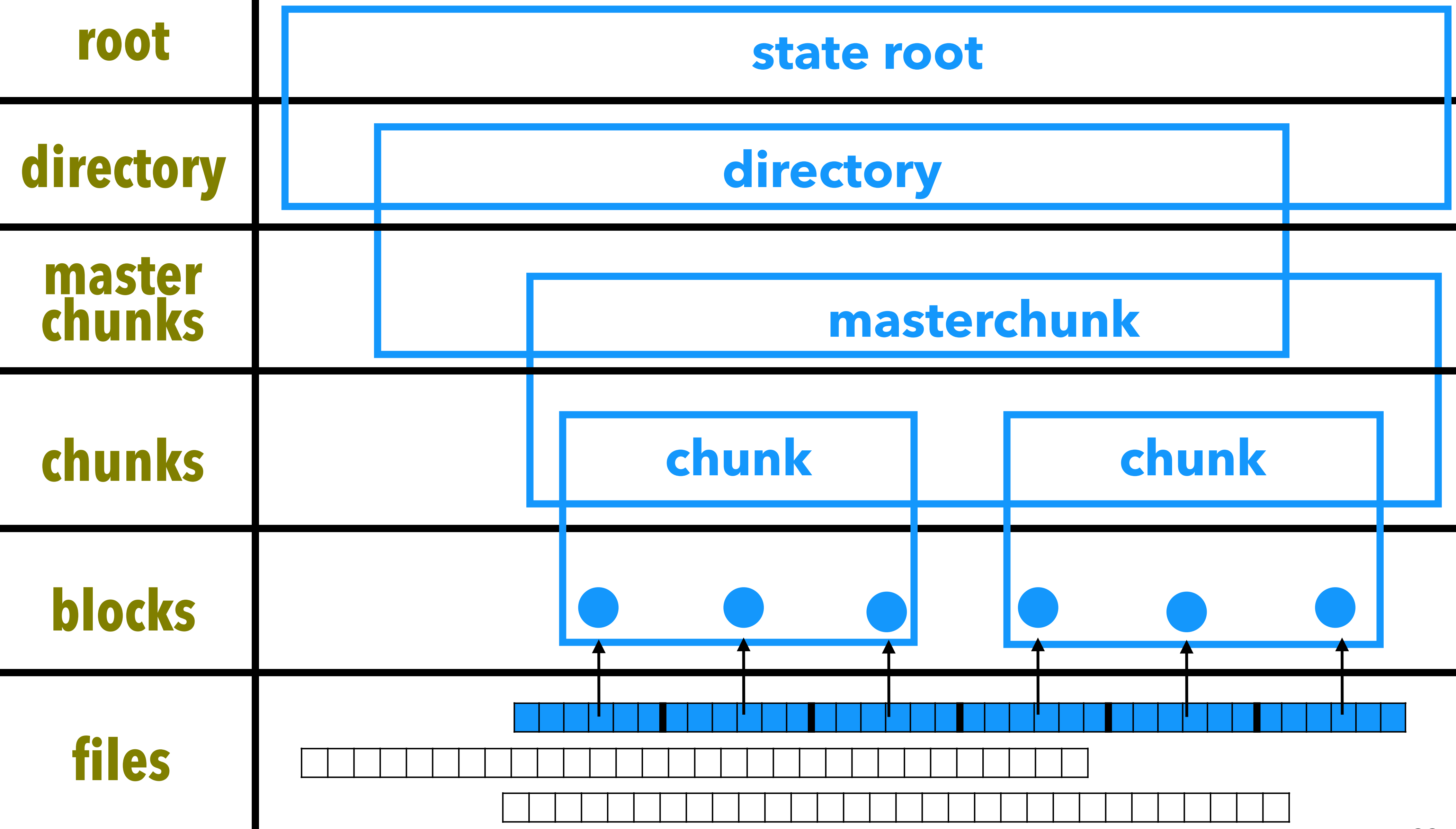


Made for:

- Incremental validation as data is loaded
- Fast verification
- Single hash tree is unsuitable

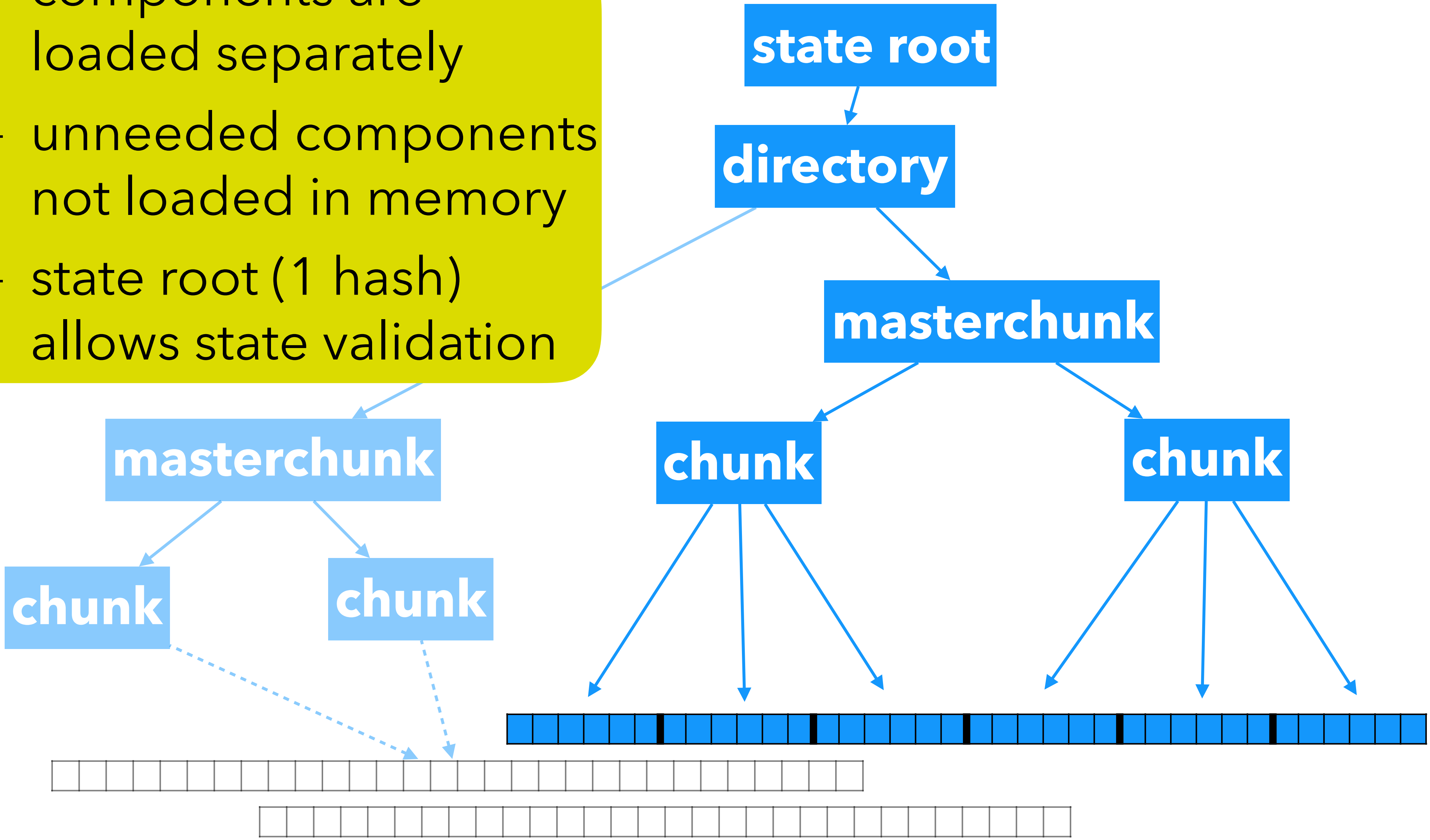


State Hierarchy



State Hierarchy

- components are loaded separately
- unneeded components not loaded in memory
- state root (1 hash) allows state validation



state root

directory

masterchunk

chunk

chunk

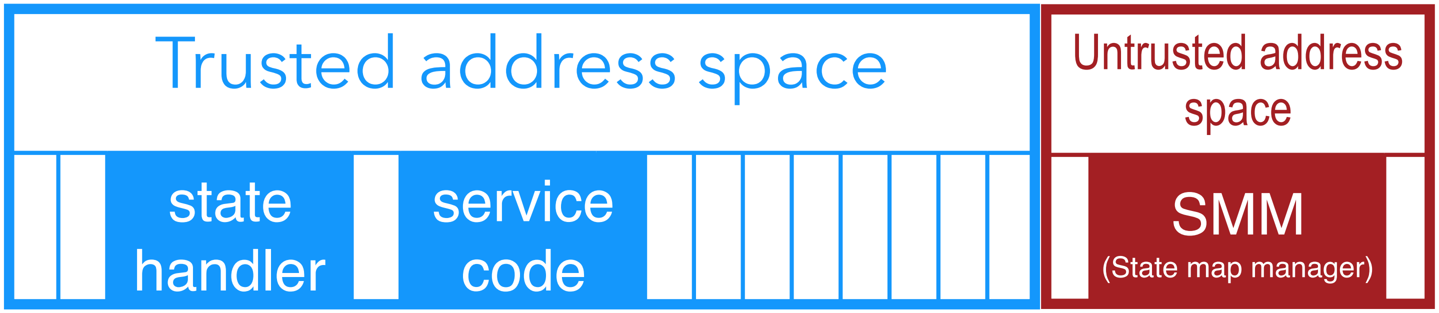
masterchunk

chunk

chunk

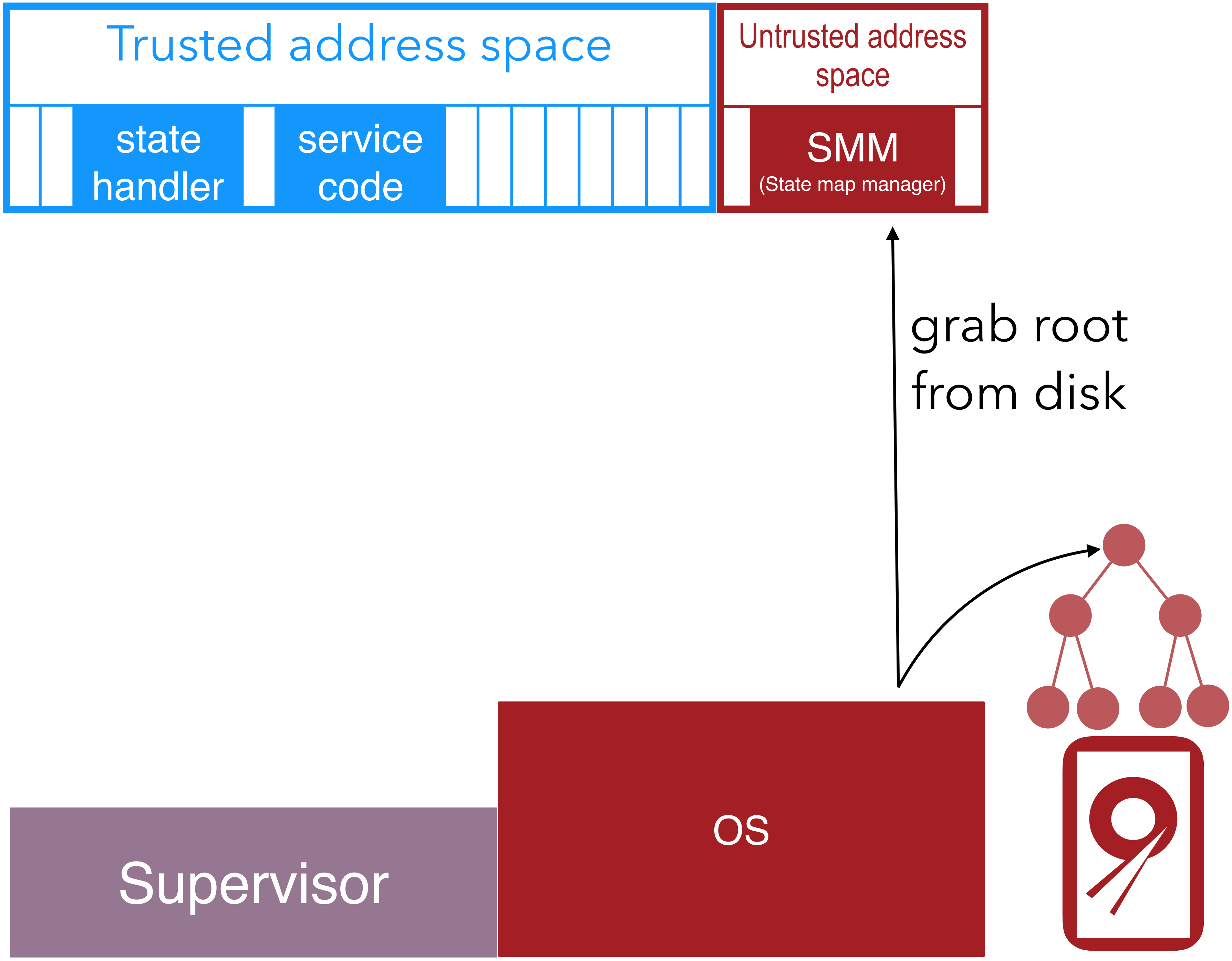
- State registration

2



When the trusted execution environment is created, only the code is available inside

Supervisor
OS



Trusted address space

state handler

service code

Untrusted address space

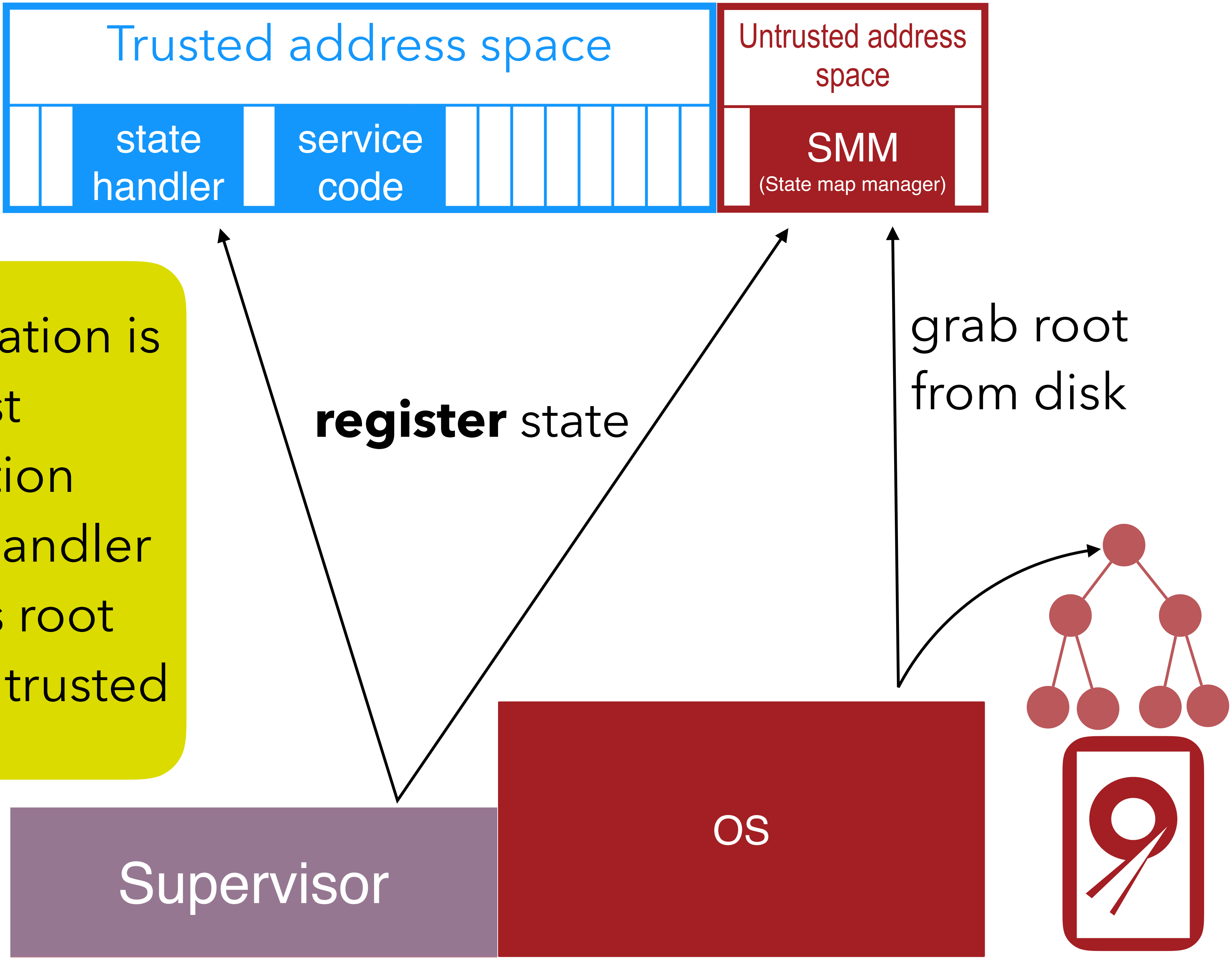
SMM

(State map manager)

grab root from disk

Supervisor

OS



Trusted address space

state handler

service code

Untrusted address space

SMM

(State map manager)

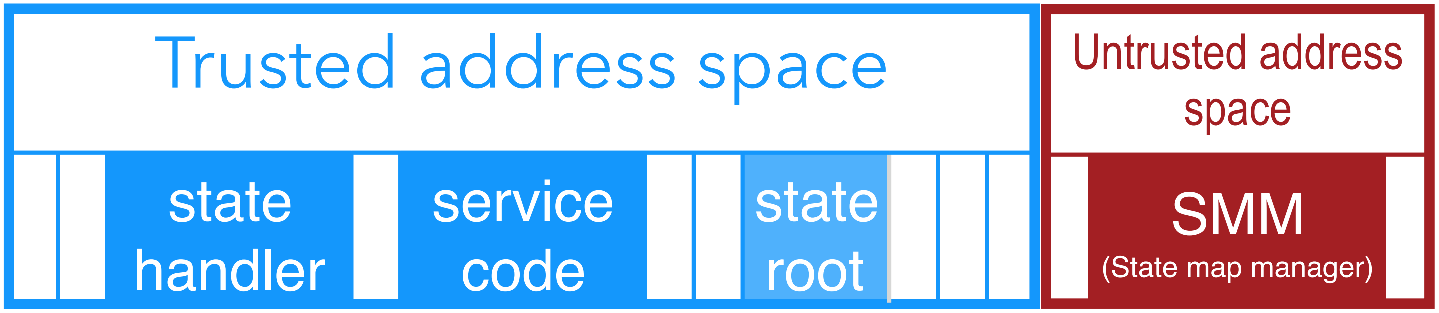
register state

grab root from disk

Supervisor

OS

- registration is the first execution
- state handler installs root
- root is trusted



Trusted address space

state handler

service code

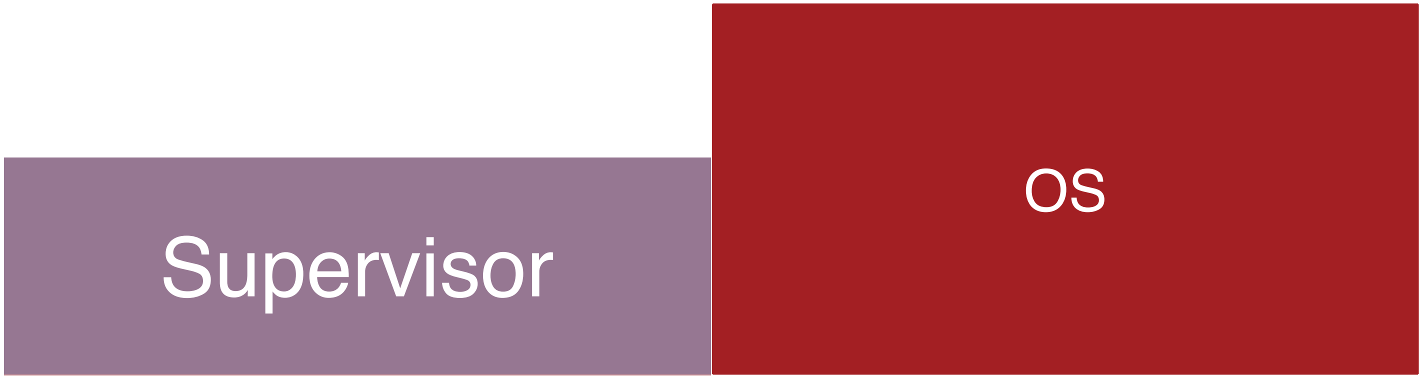
state root

Untrusted address space

SMM

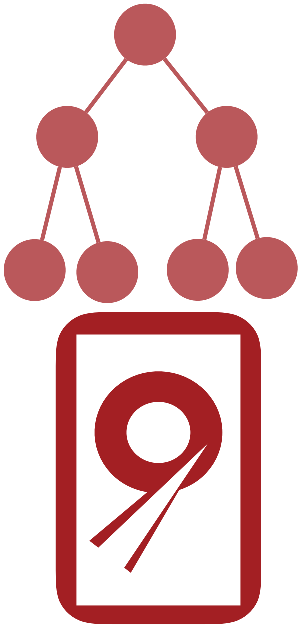
(State map manager)

- state root is available before service code runs



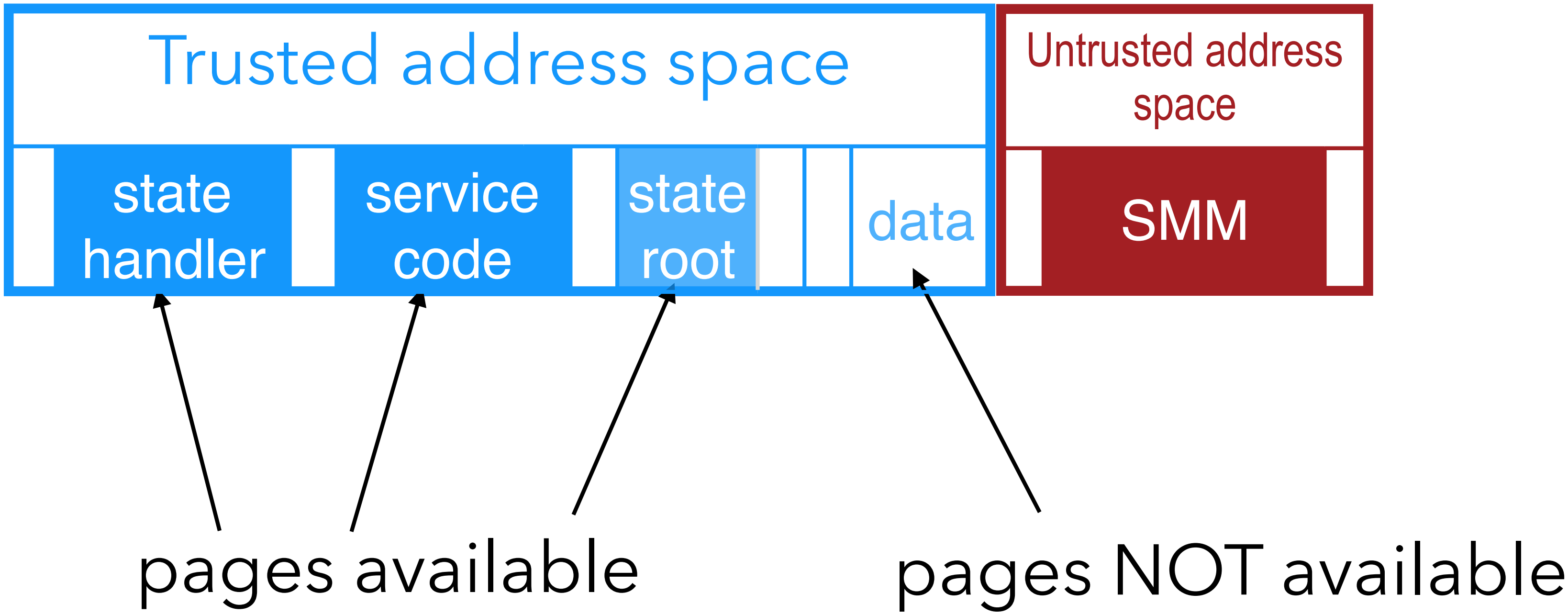
Supervisor

OS

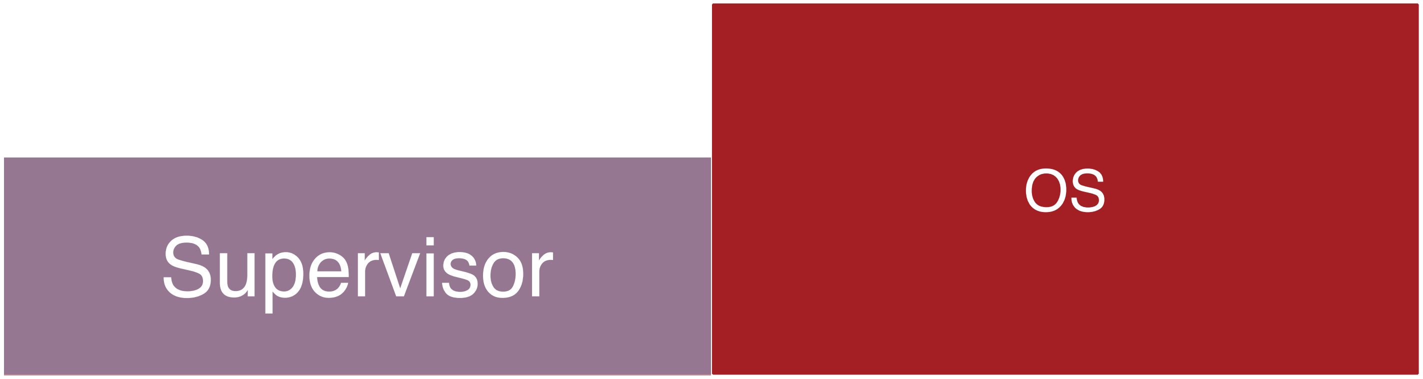


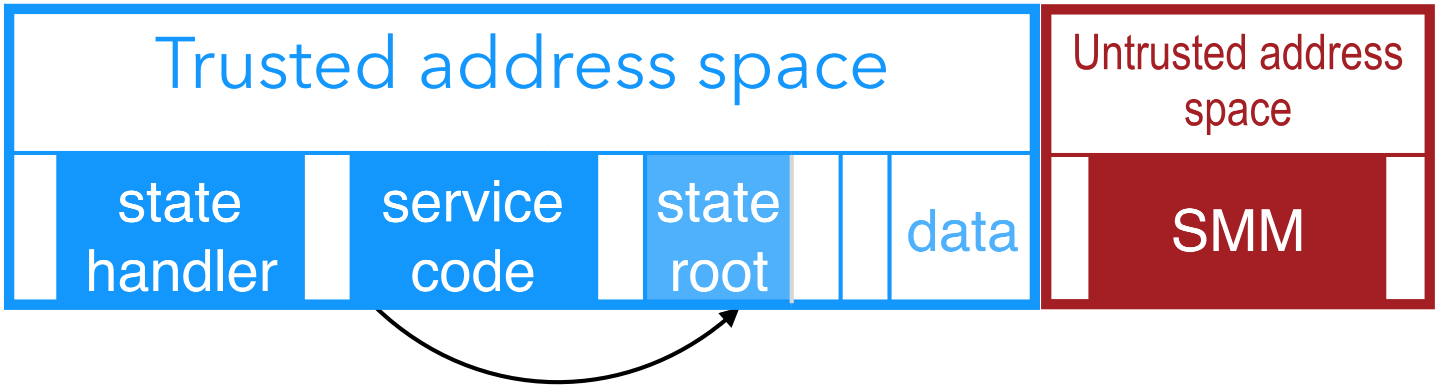
- **Data processing**

3



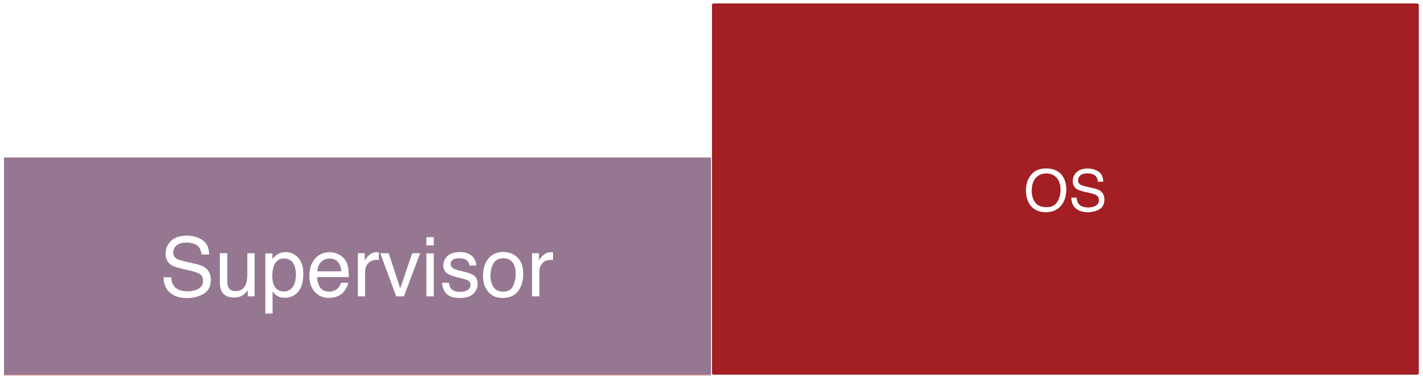
- service code has view of entire state
- state not readily available: inefficient loading it upfront

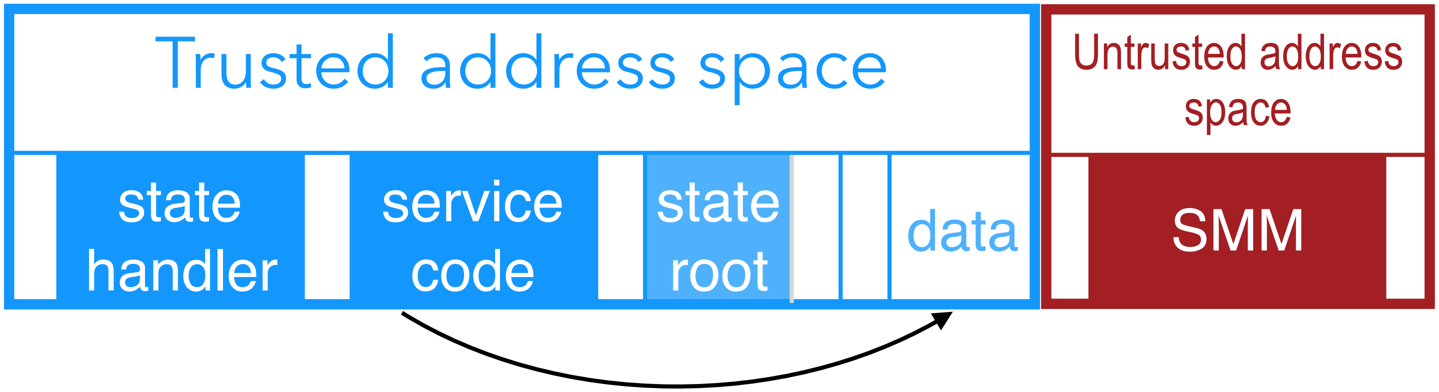




page hit on access

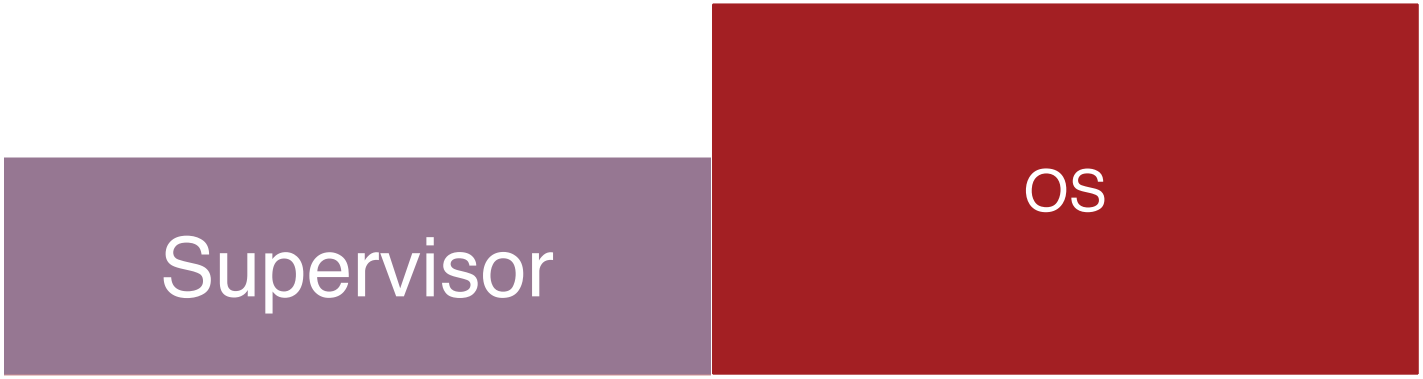
- Service code execution begins
- Service accesses data in memory
- Data retrieval is fast if data is already available

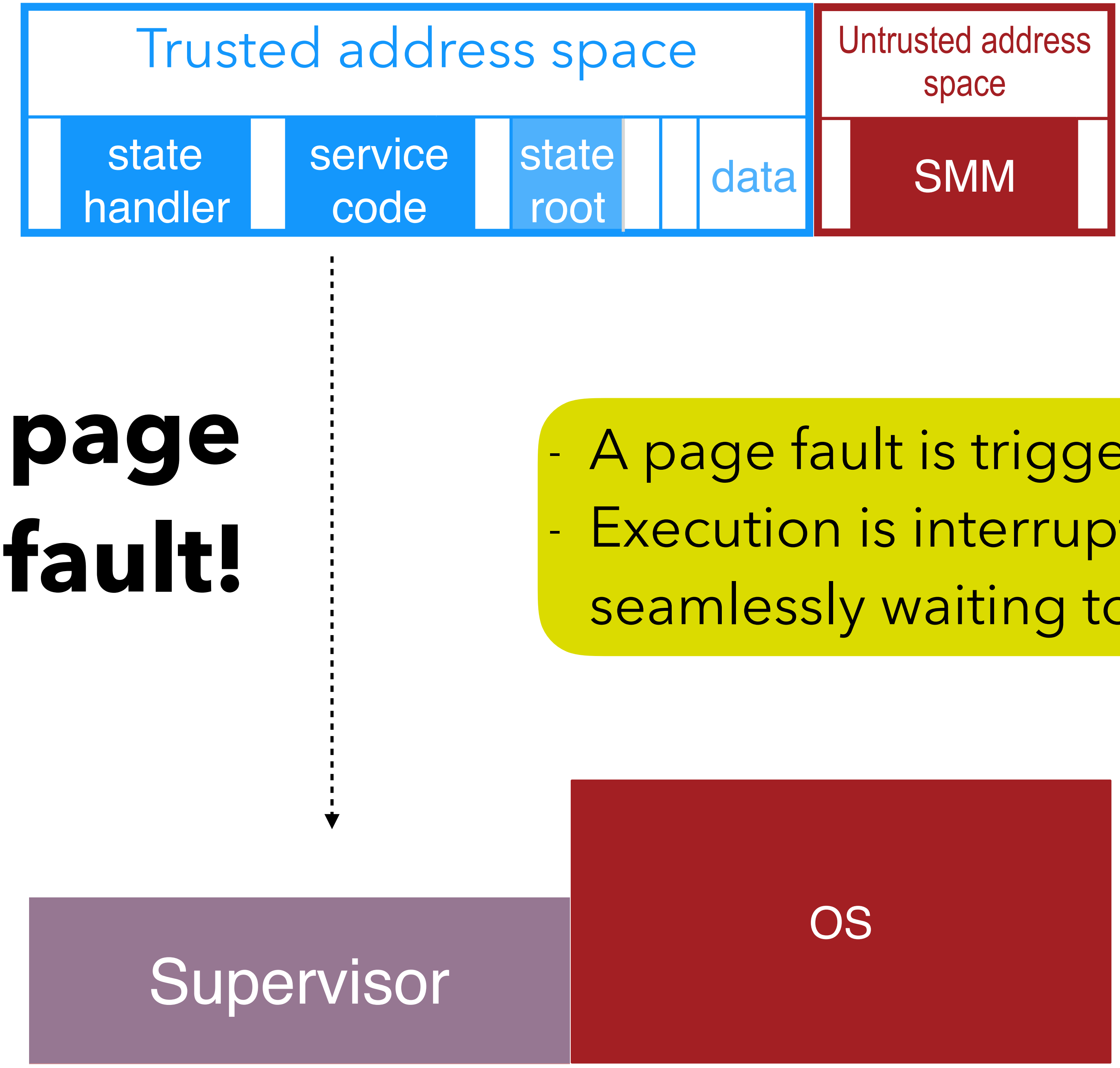




page miss on access

- Service code may access data on missing pages



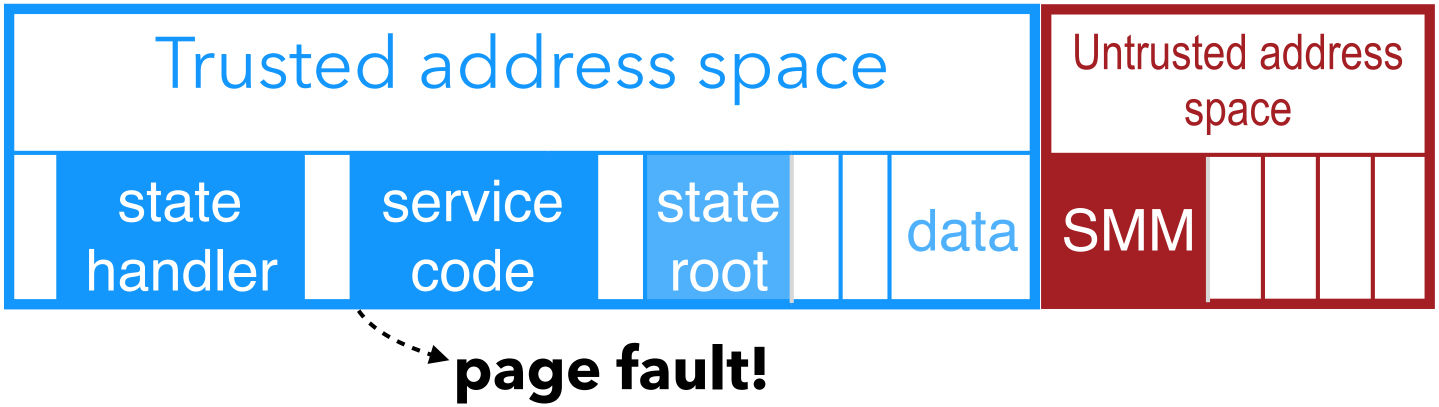


page fault!

- A page fault is triggered
- Execution is interrupted, seamlessly waiting to continue

4

- Lazy loading from memory & disk



Trusted address space

state handler

service code

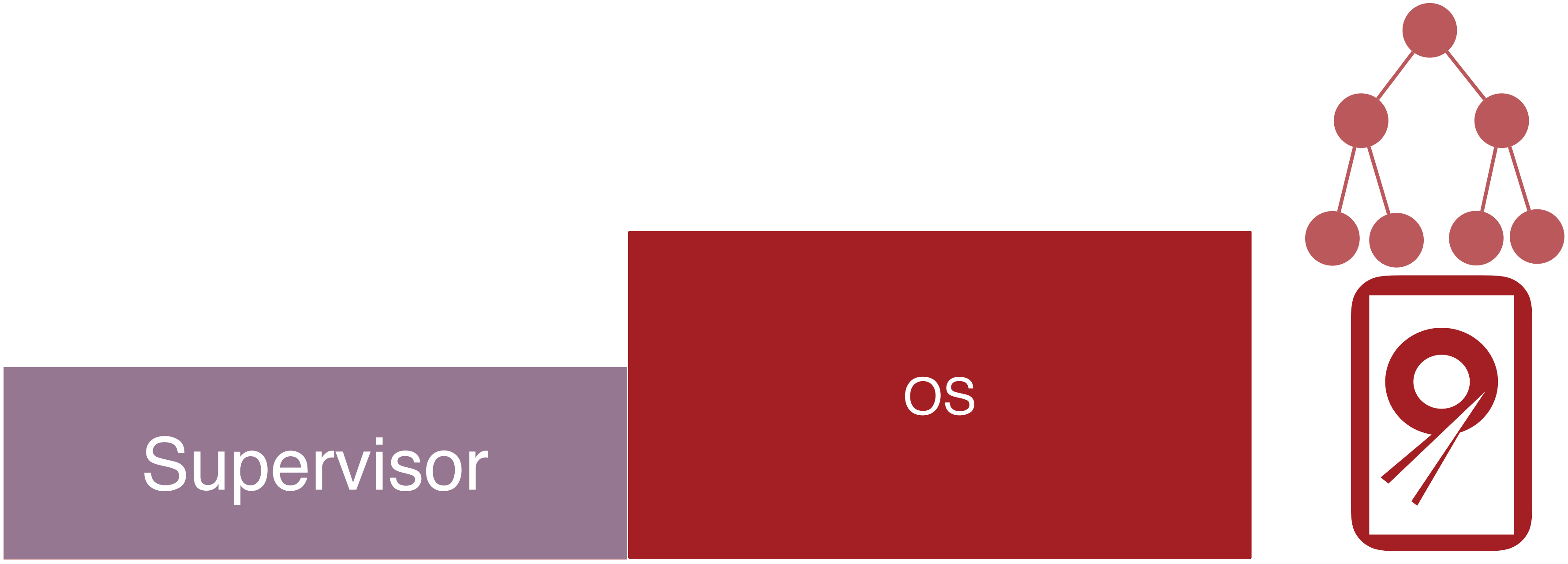
state root

data

Untrusted address space

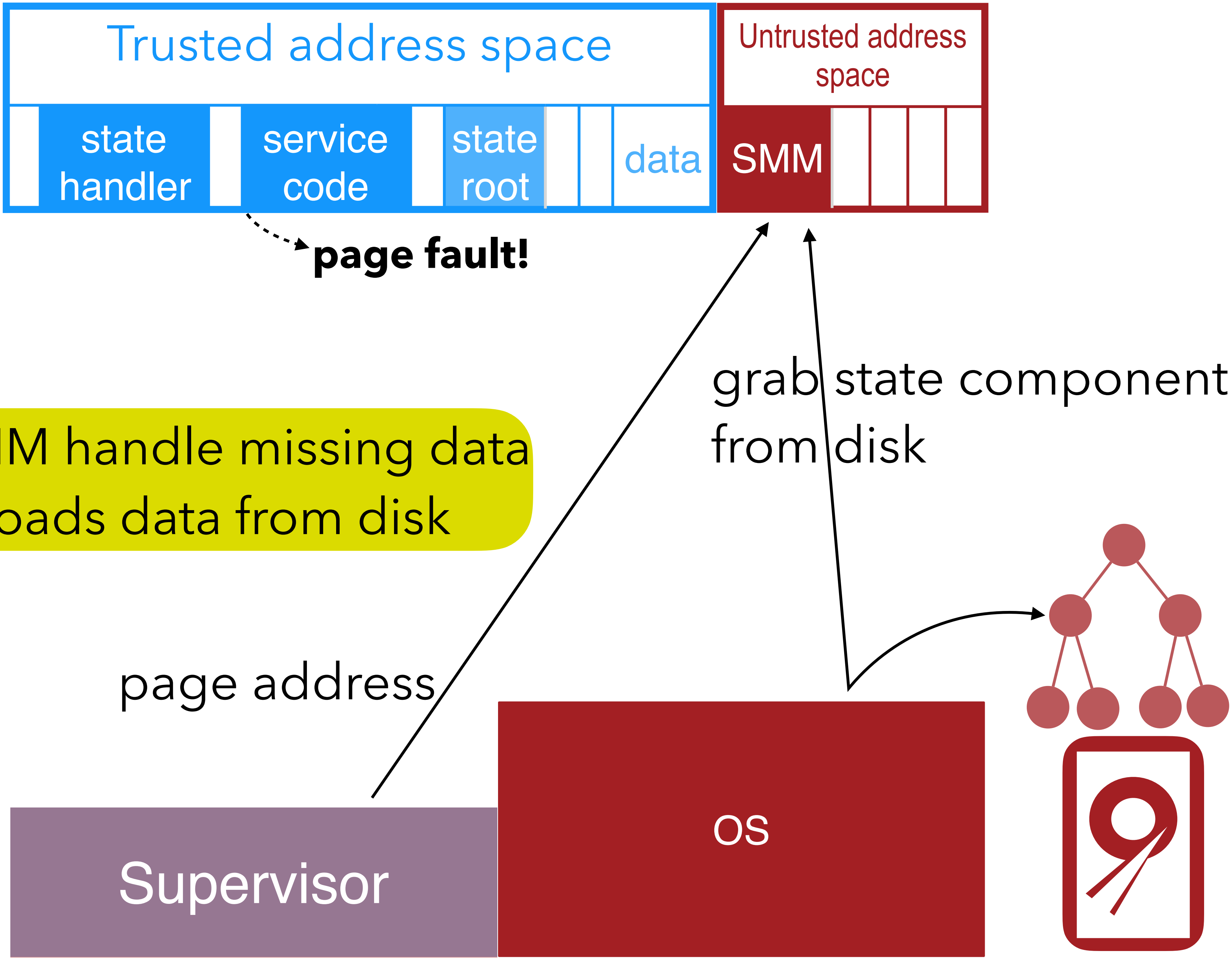
SMM

page fault!

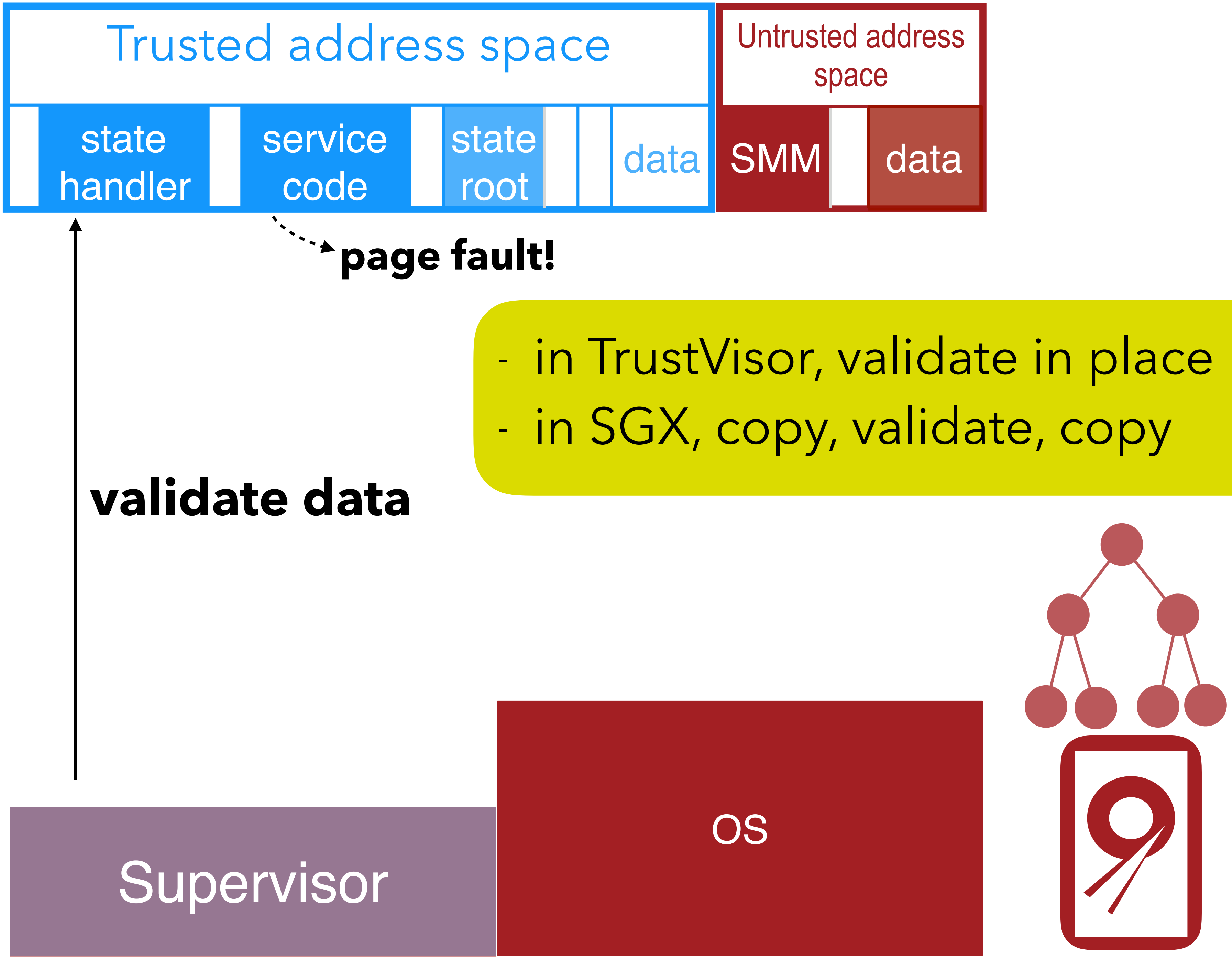


Supervisor

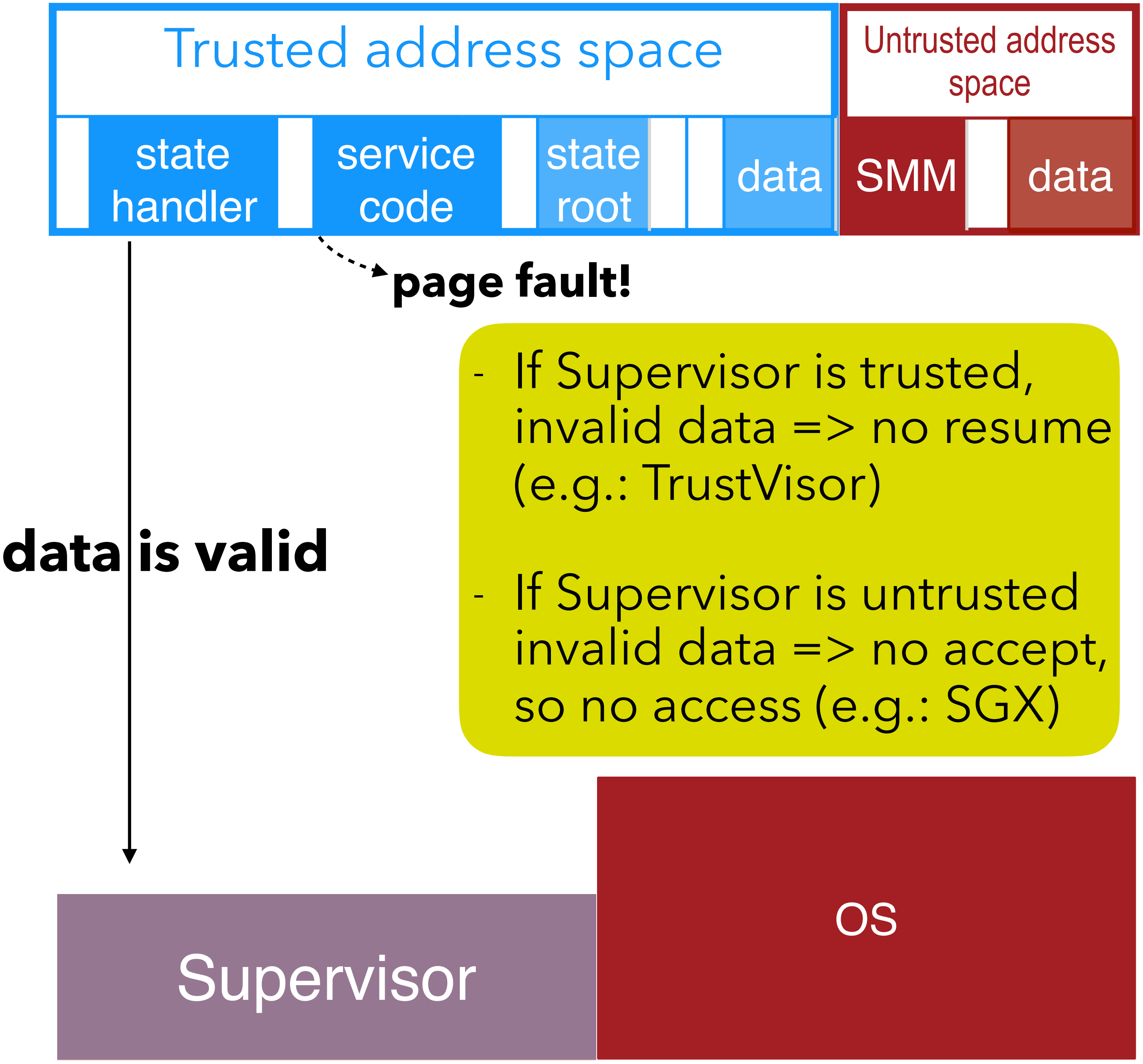
OS



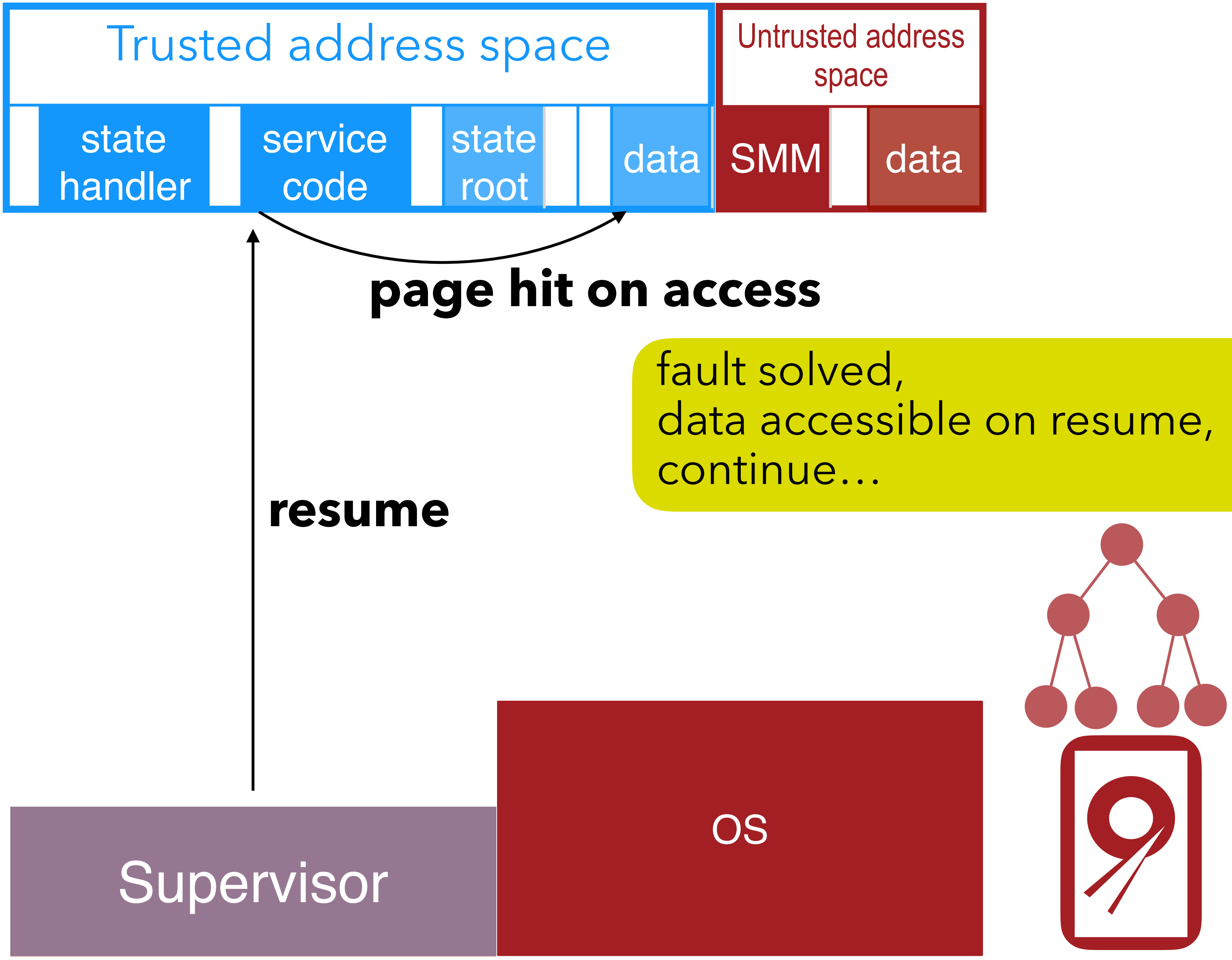
- Let SMM handle missing data
- SMM loads data from disk



- in TrustVisor, validate in place
- in SGX, copy, validate, copy



- If Supervisor is trusted, invalid data => no resume (e.g.: TrustVisor)
- If Supervisor is untrusted invalid data => no accept, so no access (e.g.: SGX)



fault solved,
data accessible on resume,
continue…

- HW-based attestation of code identity, including input request, state root, output reply, nonce
- Client checks validity of attestation and intended identities/hashes

5

- **~~Execution verification~~**

Outline

- **Evaluation**

TCB size

SGX-based

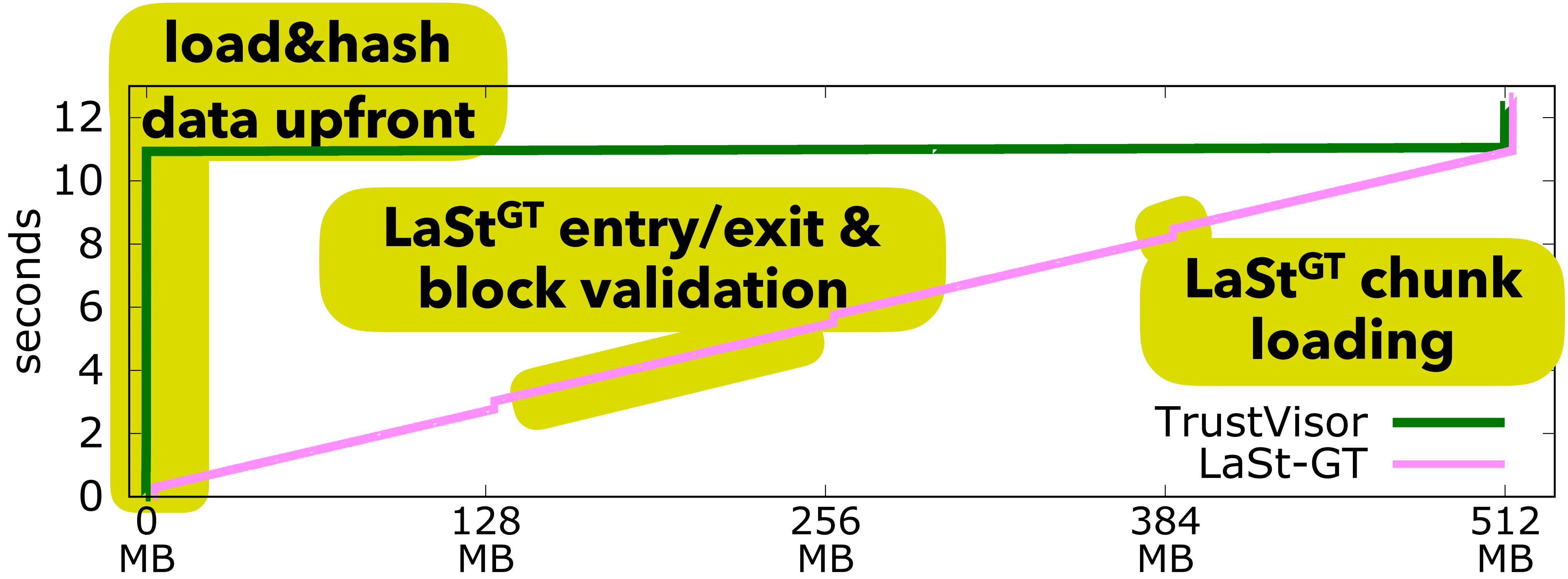
TPM/TXT based

	VC^3	Haven	LaSt^{GT}		
			hypervisor	library	SQLite (example)
KSLoC (lines of code x 1000)	9.2	O(10^3)	17	7.7	92.6

library is small compared to real service

Comparison

XMHF-TrustVisor vs. LaSt^{GT}



load&hash data upfront

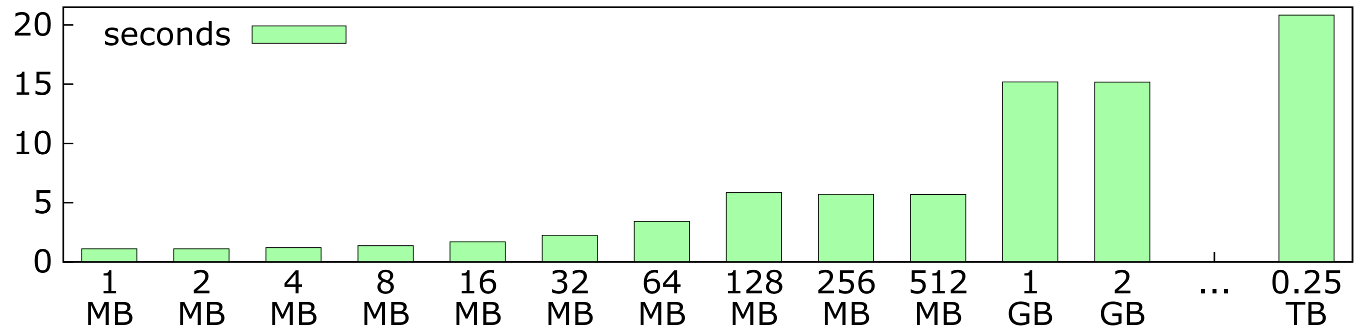
LaSt^{GT} entry/exit & block validation

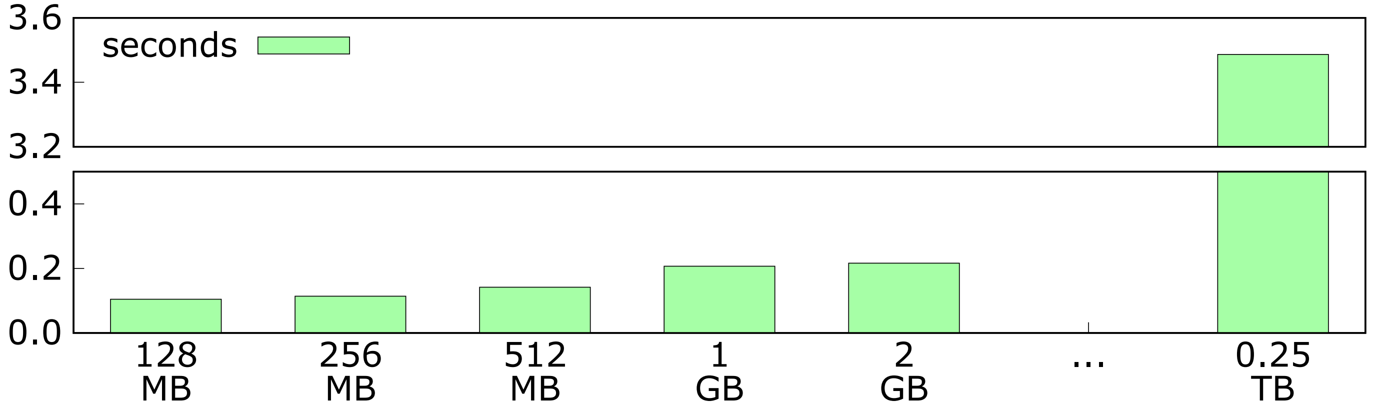
LaSt^{GT} chunk loading

LaSt^{GT} is Incremental, Faster & Scalable

SQLite on LaSt^{GT}

- First large-scale experiment on hypervisor
- Data I/O can be optimized through state hierarchy
- SGX expected to improve substantially





Conclusions

- Security for large-scale data processing can be guaranteed with a small TCB

- Virtual memory-based data handling

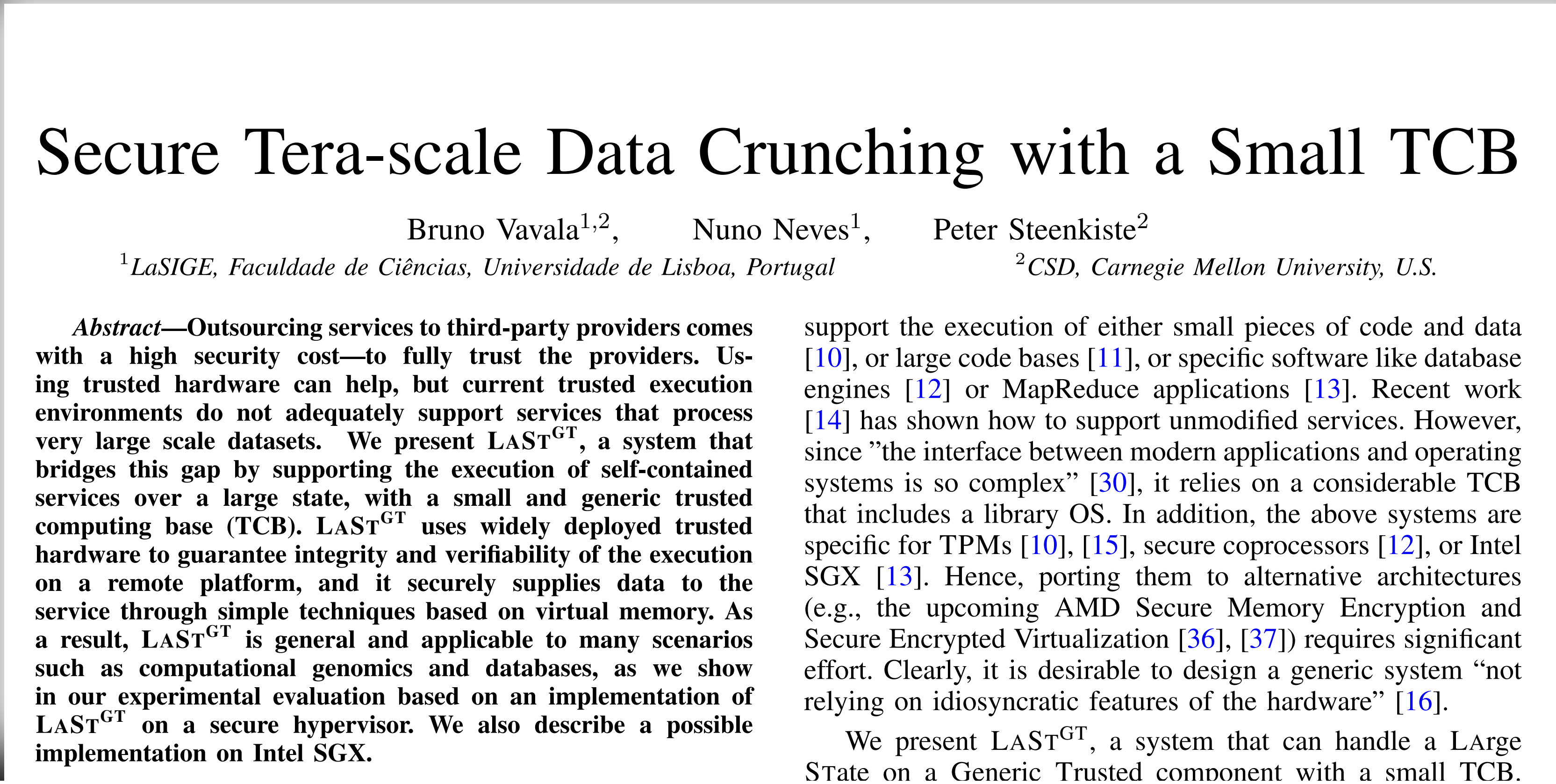
  => zero interface

- No change to source code

  => easy integration

- One design can fit diverse HW & SW

ad maiora.



Secure Tera-scale Data Crunching with a Small TCB

Bruno Vavala[1,2], Nuno Neves[1], Peter Steenkiste[2]

[1] *LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal* [2] *CSD, Carnegie Mellon University, U.S.*

***Abstract*—Outsourcing services to third-party providers comes with a high security cost—to fully trust the providers. Using trusted hardware can help, but current trusted execution environments do not adequately support services that process very large scale datasets. We present LAST^{GT}, a system that bridges this gap by supporting the execution of self-contained services over a large state, with a small and generic trusted computing base (TCB). LAST^{GT} uses widely deployed trusted hardware to guarantee integrity and verifiability of the execution on a remote platform, and it securely supplies data to the service through simple techniques based on virtual memory. As a result, LAST^{GT} is general and applicable to many scenarios such as computational genomics and databases, as we show in our experimental evaluation based on an implementation of LAST^{GT} on a secure hypervisor. We also describe a possible implementation on Intel SGX.**

support the execution of either small pieces of code and data [10], or large code bases [11], or specific software like database engines [12] or MapReduce applications [13]. Recent work [14] has shown how to support unmodified services. However, since "the interface between modern applications and operating systems is so complex" [30], it relies on a considerable TCB that includes a library OS. In addition, the above systems are specific for TPMs [10], [15], secure coprocessors [12], or Intel SGX [13]. Hence, porting them to alternative architectures (e.g., the upcoming AMD Secure Memory Encryption and Secure Encrypted Virtualization [36], [37]) requires significant effort. Clearly, it is desirable to design a generic system "not relying on idiosyncratic features of the hardware" [16].

We present LAST^{GT}, a system that can handle a LArge STate on a Generic Trusted component with a small TCB.

(blank)

“Never trust a computer you can’t throw out the window.”

Steve Wozniak

“No computer system can be absolutely secure.”

(excerpt from)

Intel’s Legal Desclaimer