Private Cloud Computing

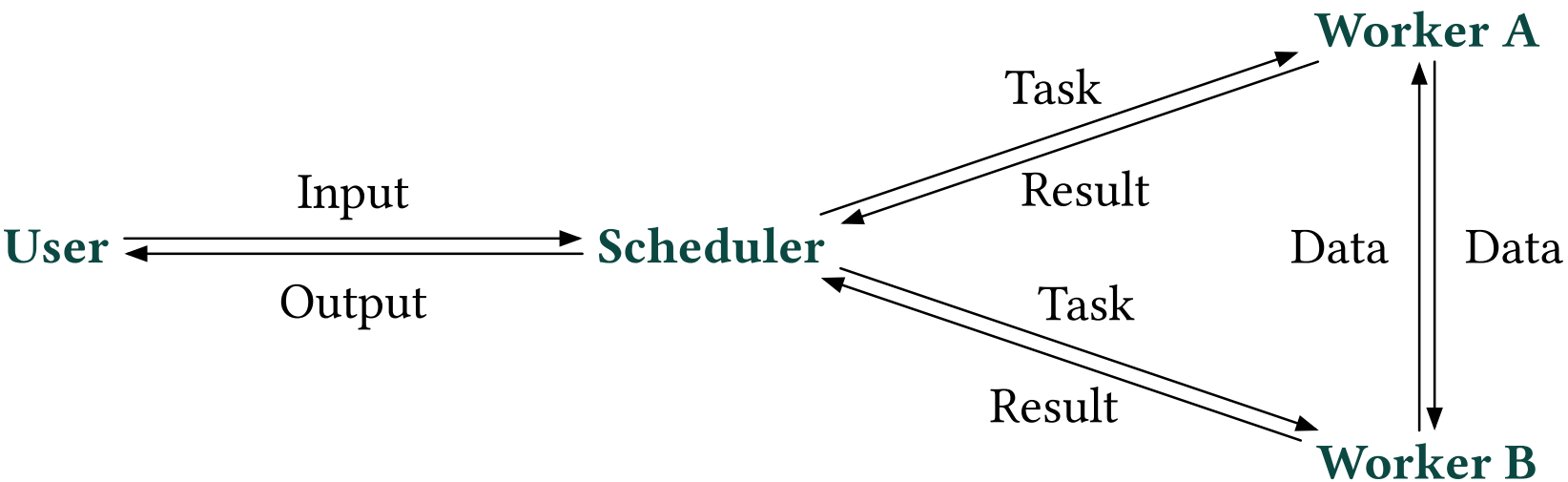
Jiale Guan

2024-08-07

Outline

Outline

1.1 Architecture



Outline

2.1 Taxonomy

Requirements	Threat	Guarantees
Stateless computation	Trace of data after processing e.g. Logging, debugging	(Purpose) Only use user data to perform requested operations (Transient) Delete the data after fulfilling the request (Scope) Not available to even Apple staff
Enforceable guarantees	Technical enforceability e.g. External TLS-terminating load balancer	(Hardware) Secure Enclave, Secure Boot (System) Signed System Volume, Swift on Server (Software) Code Signing, Sandboxing
No privileged runtime access	Privileged interfaces e.g. Shell access by SREs	No remote shell. Only pre-specified, structured, and audited logs/metrics can leave the node User data is reviewed by multiple indepedent layers
Non-targetability	Targeted attack e.g. Steer request to compromised nodes	(Hardware) Hardened supply chain (Scheduler) Requests cannot be user/content-specific routed (Anonymity) OHTTP Relay, RSA Blind Signature (Scope) No system-wide encryption
Verifiable transparency	Uninspected code	Every production build of PCC publicly available

2.2 Requirements

Stateless computation

Private Cloud Compute must use the personal user data that it receives exclusively for the purpose of fulfilling the user's request. This data must never be available to anyone other than the user, not even to Apple staff, not even during active processing. And **this data must not be retained**, including via logging or for debugging, after the response is returned to the user. In other words, we want a strong form of stateless data processing where **personal data leaves no trace** in the PCC system.

Enforceable guarantees

Security and privacy guarantees are strongest when they are entirely technically enforceable, which means it must be possible to **constrain and analyze all the components** that critically contribute to the guarantees of the overall Private Cloud Compute system. To use our example from earlier, it's very difficult to reason about what a TLS-terminating load balancer may do with user data during a debugging session. Therefore, PCC must not depend on such external components for its core security and privacy guarantees. Similarly, operational requirements such as collecting server metrics and error logs must be supported with mechanisms that do not undermine privacy protections.

No privileged runtime access

Private Cloud Compute must not contain privileged interfaces that would enable Apple's site reliability staff to bypass PCC privacy guarantees, even when working to resolve an outage or other severe incident. This also means that PCC must not support a mechanism by which the privileged access envelope could be enlarged at runtime, such as by loading additional software.

Non-targetability

An attacker should not be able to attempt to compromise personal data that belongs to specific, targeted Private Cloud Compute users without attempting a broad compromise of the entire PCC system. This must hold true even for exceptionally sophisticated attackers who can attempt physical attacks on PCC nodes in the supply chain or attempt to obtain malicious access to PCC data centers. In other words, a limited PCC compromise must not allow the attacker to **steer requests from specific users to compromised nodes**; targeting users should require a wide attack that's likely to be detected. To understand this more intuitively, contrast it with a traditional cloud service design where every application server is provisioned with database credentials for the entire application database, so a compromise of a single application server is sufficient to access any user's data, even if that user doesn't have any active sessions with the compromised application server.

Verifiable transparency

Security researchers need to be able to verify, with a high degree of confidence, that our privacy and security guarantees for Private Cloud Compute match our public promises. We already have an earlier requirement for our guarantees to be enforceable. Hypothetically, then, if security researchers had sufficient access to the system, they would be able to verify the guarantees. But this last requirement, verifiable transparency, goes one step further and does away with the hypothetical: security researchers must be able to verify the security and privacy guarantees of Private Cloud Compute, and they must be able to verify that the software that's running in the PCC production environment is the same as the software they inspected when verifying the guarantees.

Outline

3.1 Optimization Goals

- Lower resource requirements
  - Model compression
  - Collaborative inference
  - Offloading to CPU and disk
- Higher throughput
  - Batch
  - Tensor parallelism
  - Sequence parallelism (Input sequence)
  - Pipeline parallelism (Stage placement)
- Lower latency
  -

3.2 Violations

Requirement	Violations
Stateless computation	Logging, prioritization, history metadata
Enforceable guarantees	Data transfer/duplication, data offloading, access control
No privileged runtime access	Monitoring, debugging, profiling
Non-targetability	Biased scheduler, input/output leakage
Verifiable transparency	Uninspected code

Universal problems: Access control between worker nodes.

Requirement	Violations	FT 22	FlexGen 23	Sarathi 23	Mooncake 24
Stateless computation	Logging				
	History metadata				
Enforceable guarantees	Data transfer/duplication				✓
	Data offloading		✓		✓
No privileged runtime access	Monitoring				
	Debugging				
	Offline Profiling				✓
Non-targetability	Priority-based scheduler				✓ length
	Input/output leakage				
Verifiable transparency	Uninspected code				✓

3.3 Optimizations

S: Stateless computation **E**: Enforceable guarantees **P**: No privileged runtime access **T**: Non-targetability **V**: Verifiable transparency

Type	Optimization	Threat	FT 22	Orca 22	vLLM 23	FlexGen 23	Sarathi 23	DistServe 24	Mooncake Moonshot	DeepSpeed Microsoft	TensorRT NVIDIA
Storage (KVCache)	Paging				✓1st		✓		✓	✓	
	Duplication								✓		
	Offloading	**SE**				✓			✓		
	Pulling	**SET**						✓			
	Model Compression										
Scheduler	Priority-Based	**T**							✓	✓	
	Online/Offline Profiling	**P**						✓	✓		
	Local Scheduler	**ET**							✓		
	Disaggregated Arch							✓	✓		
PP	Iteration-Level Batch			✓1st	✓		✓	✓	✓	✓	
	Chunked Prefill						✓1st		✓	✓	
	Prepack Prefill							✓			
Verification	Uninspected Code	**V**							✓		

Production LLMs

Server	Vendor	Website
-LLM	NVIDIA	`https://github.com/NVIDIA/TensorRT-LLM?tab=readme-ov-file`

TorchServe Triton HuggingFace TGI FastServe

DistServe TetriInfer

Prompt Cache SGLang AttentionStore Preble

Pollux

Refer to DistServe Related Work

Thanks

Outline

Appendix

https://github.com/microsoft/DeepSpeed/tree/master/blogs/deepspeed-fastgen