Towards Efficient End-to-End Encryption for Container Checkpointing Systems

Radostin Stoyanov - PhD Student, Scientific Computing Group

Adrian Reber, Daiki Ueno, Michał Cłapiński, Andrei Vagin

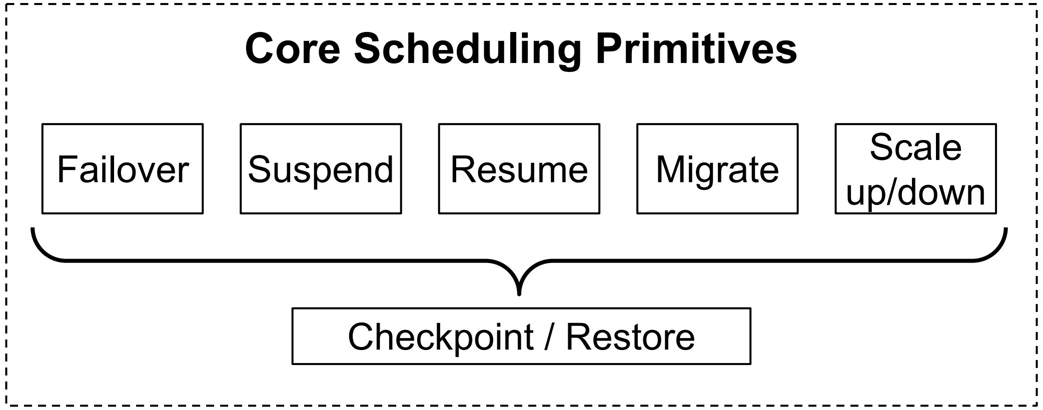
Supervisor: Prof. Rodrigo Bruno, Prof. Wes Armour

Container Checkpointing Use Cases

Understanding the use cases and mechanisms for checkpoint/restore

Container Checkpointing Use Cases

- Fault-tolerance [1, 5]
- Fast application start-up [2, 6]
- Preemptive scheduling [5, 8]
- Load balancing (job migration) [7, 8]
- Forensic analysis [3, 4]



[1] Tanmaey Gupta, et al. *“Just-In-Time Checkpointing: Low Cost Error Recovery from Deep Learning Training Failures”* (EuroSys ‘24)
[2] Sumer Kohli, et al. *“Pronghorn: Effective Checkpoint Orchestration for Serverless Hot-Starts”* (EuroSys ‘24)
[3] Adrian Reber. *“Forensic Container Checkpointing and Analysis”* (Kubernetes Community Days Zürich 2023)
[4] Daniel Simionato, et. al. *“Digital Forensics with Container Checkpointing”* (Open Source Summit Europe 2023)
[5] Dharma Shukla, et al. *"Singularity: Planet-scale, Preemptive and Elastic Scheduling of AI Workloads"* (2022)
[6] Ritesh Naik, et al. *“Container Checkpoint/Restore at Scale for Fast Pod Startup Time”* (KubeCon EU 2021)
[7] Shubham Chaudhary, et al. *“Balancing Efficiency and Fairness in Heterogeneous GPU Clusters for Deep Learning”* (EuroSys ‘20)
[8] Victor Marmol, et al. *“Task Migration at Scale Using CRIU”* (Linux Plumbers Conference 2018)

Security Risks & Challenges

Storing unencrypted checkpoint data can introduce security risks

Security Risks & Challenges

- **Security Risks**
  - Access to sensitive data (session hijacking)
  - Injecting malicious code (backdoor)
  - Altering control flow of applications (privilege escalation)

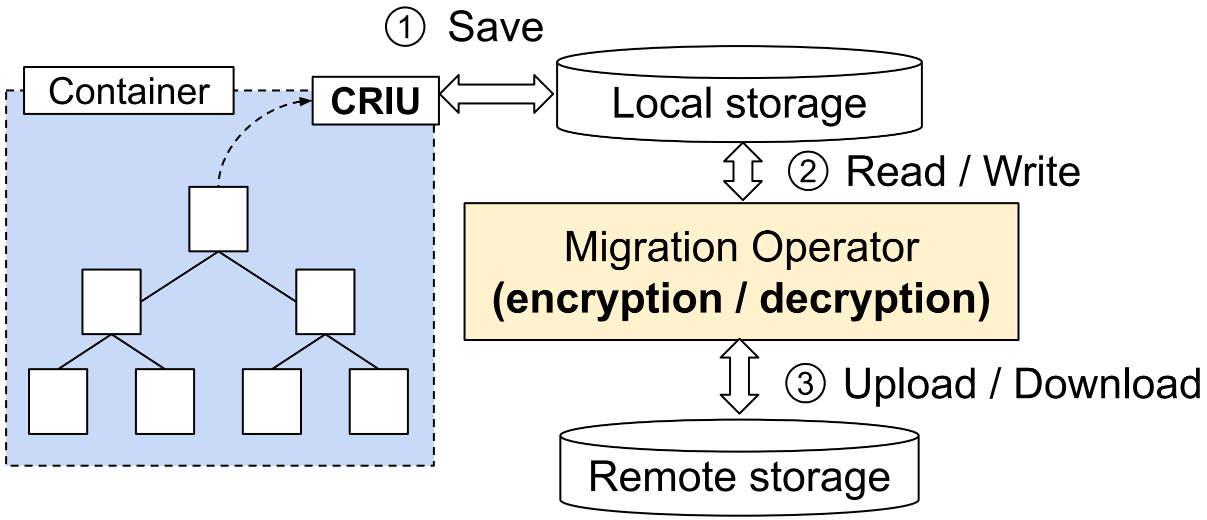
- **Challenges**
  - Performance optimizations (iterative checkpointing & memory deduplication)
  - Authentication and authorization in multi-tenant clusters
  - Verifying integrity and confidentiality of checkpoint data

Existing Encryption Methods

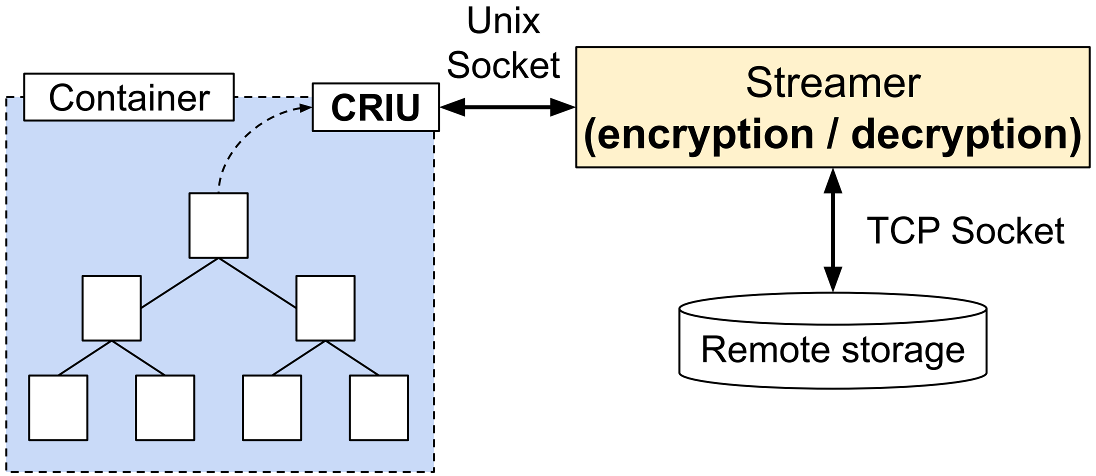
Protecting sensitive data in container checkpoints

Checkpoint Encryption Methods

Local encryption [1]



Streaming encryption [2]



[1] Victor Marmol, et al. *“Task Migration at Scale Using CRIU”* (Linux Plumbers Conference 2018)

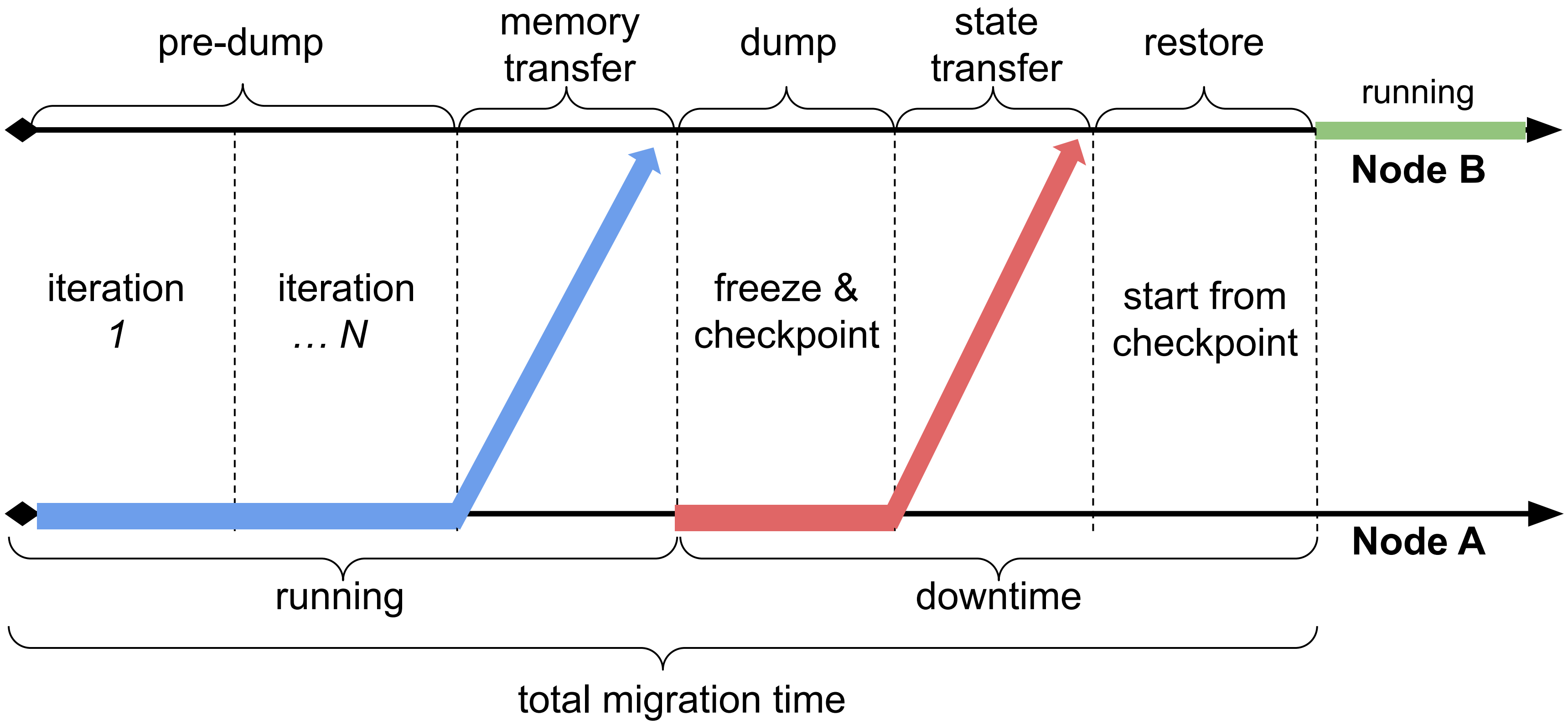
[2] Nicolas Viennot, “*Fast checkpointing with criu-image-streamer*” (Linux Plumbers Conference 2020)

Iterative Checkpointing

Enabling pre-copy live migration

Iterative Checkpointing

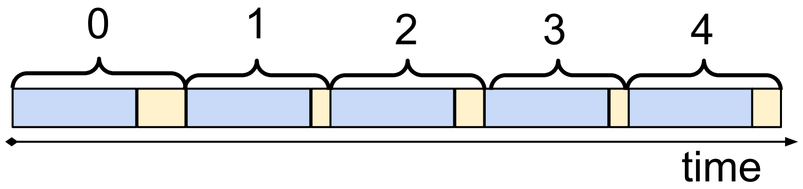
Live Migration

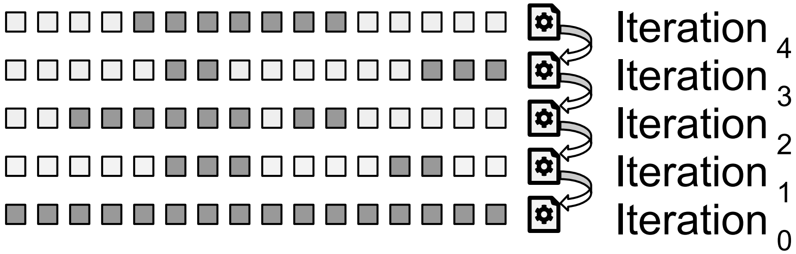


Iterative Checkpointing

Fault Tolerance

Requires multiple decryption cycles to check data availability in previous checkpoints





🟦 Computation

🟨 Checkpoint

⬜ Unmodified Memory

▨ Modified Memory

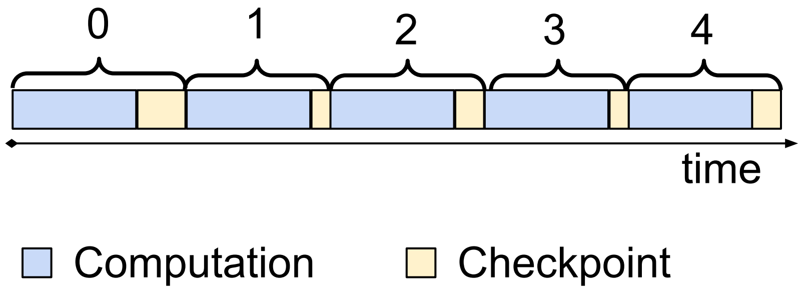
⚙ Memory Mapping

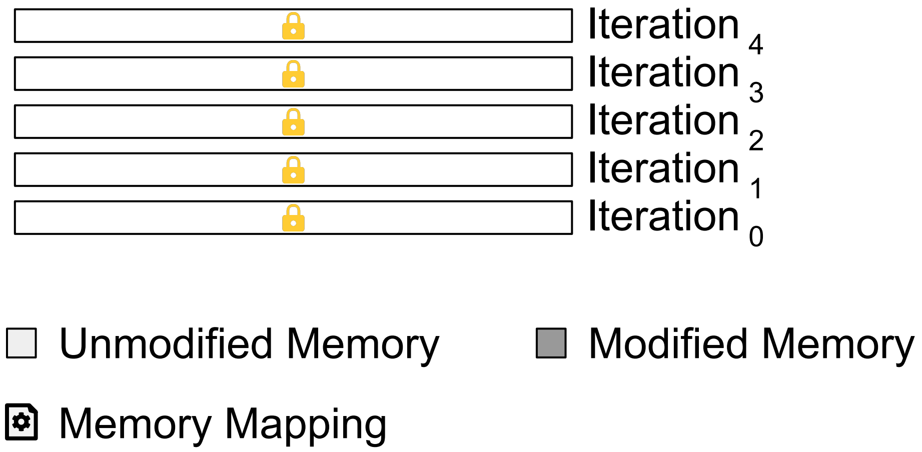
```
# Clear soft-dirty bit
$ echo 4 > /proc/${PID}/clear_refs
```

Iterative Checkpointing

Fault Tolerance

Requires multiple decryption cycles to check data availability in previous checkpoints





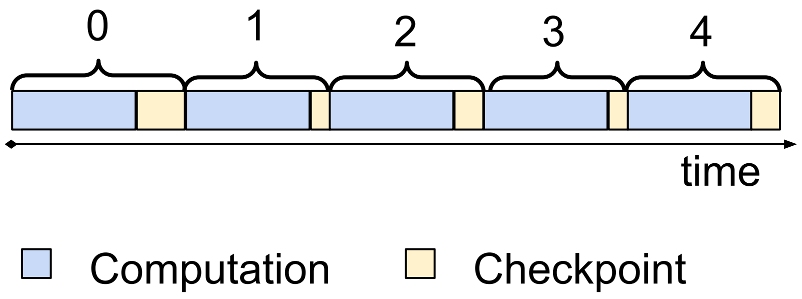
```
# Clear soft-dirty bit
$ echo 4 > /proc/${PID}/clear_refs
```

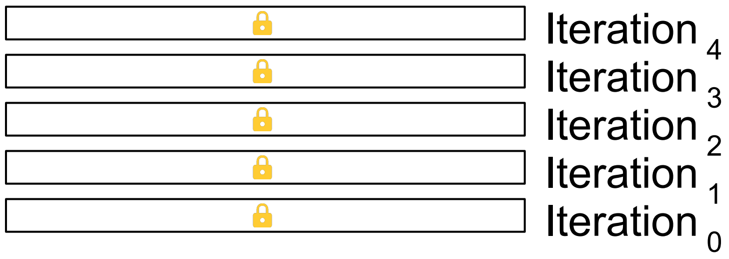
Memory Deduplication

Reducing the amount of checkpoint data

Memory Deduplication

Requires multiple rounds of *full* encryption + decryption to modify data in previous checkpoints

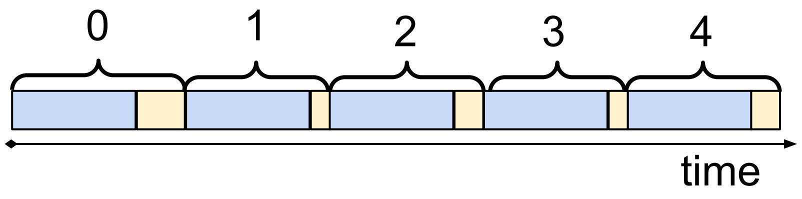




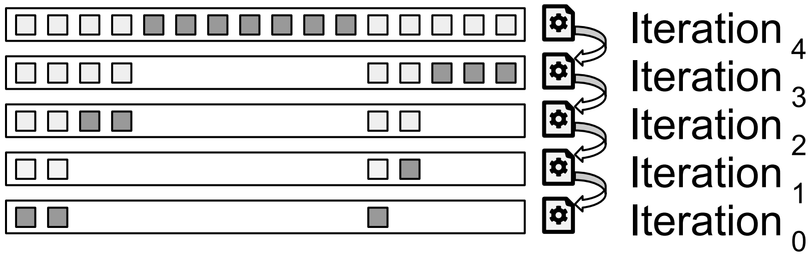
```
/* Deallocate file space */
fallocate(KEEP_SIZE|PUNCH_HOLE)
```

Memory Deduplication

Requires multiple rounds of *full* encryption + decryption to modify data in previous checkpoints



🟦 Computation 🟨 Checkpoint

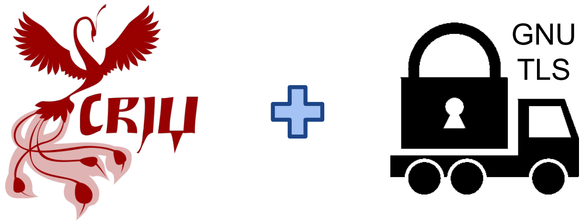


□ Unmodified Memory ■ Modified Memory

⚙ Memory Mapping

```
/* Deallocate file space */
fallocate(KEEP_SIZE|PUNCH_HOLE)
```

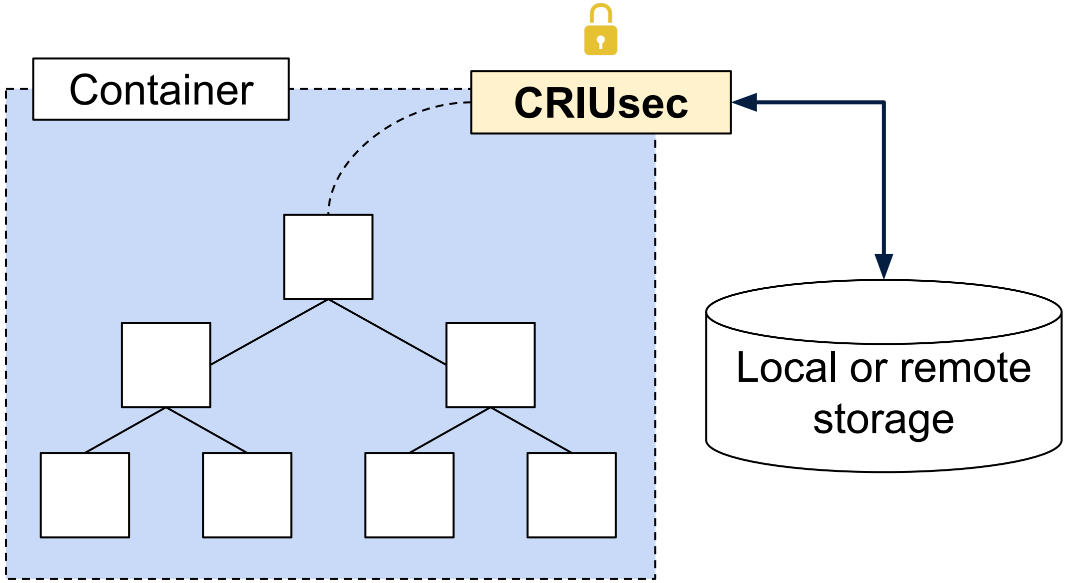
Built-in Encryption



Adding support for end-to-end checkpoint encryption

CRIUsec

CRIU with Built-in Encryption Support



End-to-end Encryption

- Checkpoint data is encrypted *before* it is saved to disk
- Decryption happens immediately *after* reading data from disk and *before* restoring the process tree

Key Management

TLS Support in CRIU

```
/etc/pki/
├── CA
│   ├── cacert.pem
│   └── cacrl.pem
└── criu
    ├── cert.pem
    └── private
        └── key.pem
```

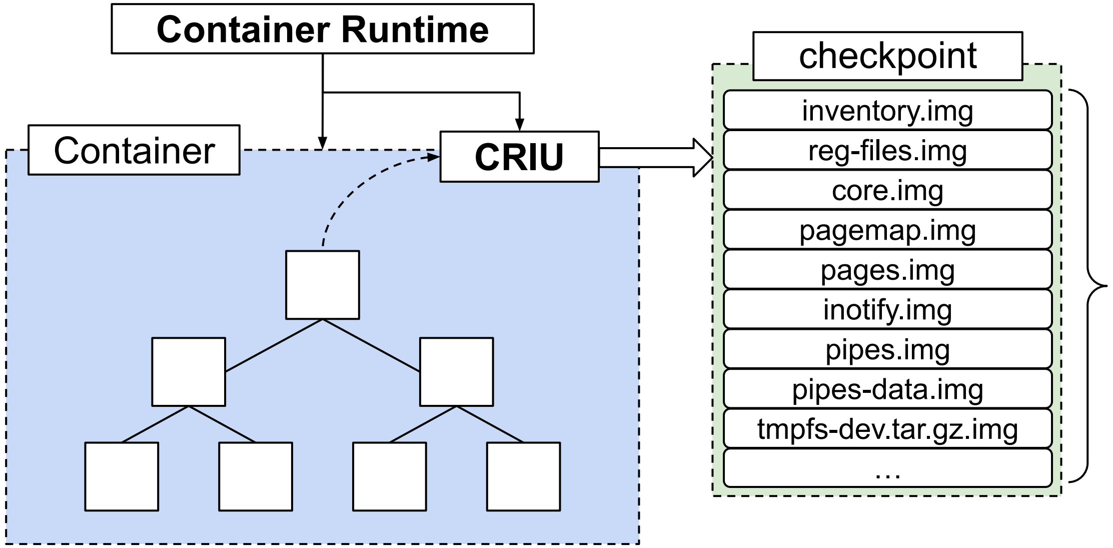
```
[dst]$ criu page-server --tls

[src]$ criu dump --tls --page-server --address <dst>
```

https://criu.org/TLS

Radostin Stoyanov, et. al. *“Secure Image-less Container Migration”* (Linux Plumbers Conference 2019)

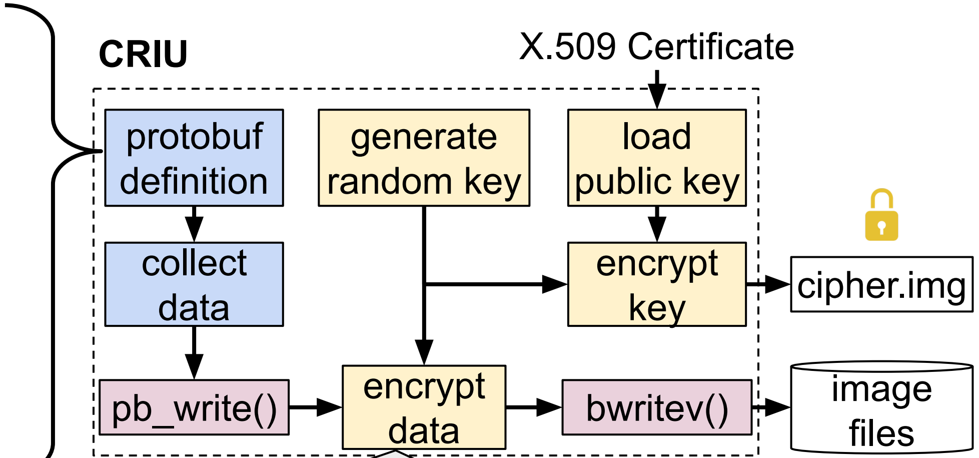
Checkpoint Images



1. **Protocol buffer format**
2. **Third-party format (raw images)**
3. **Memory pages**

Images in Protobuf Format

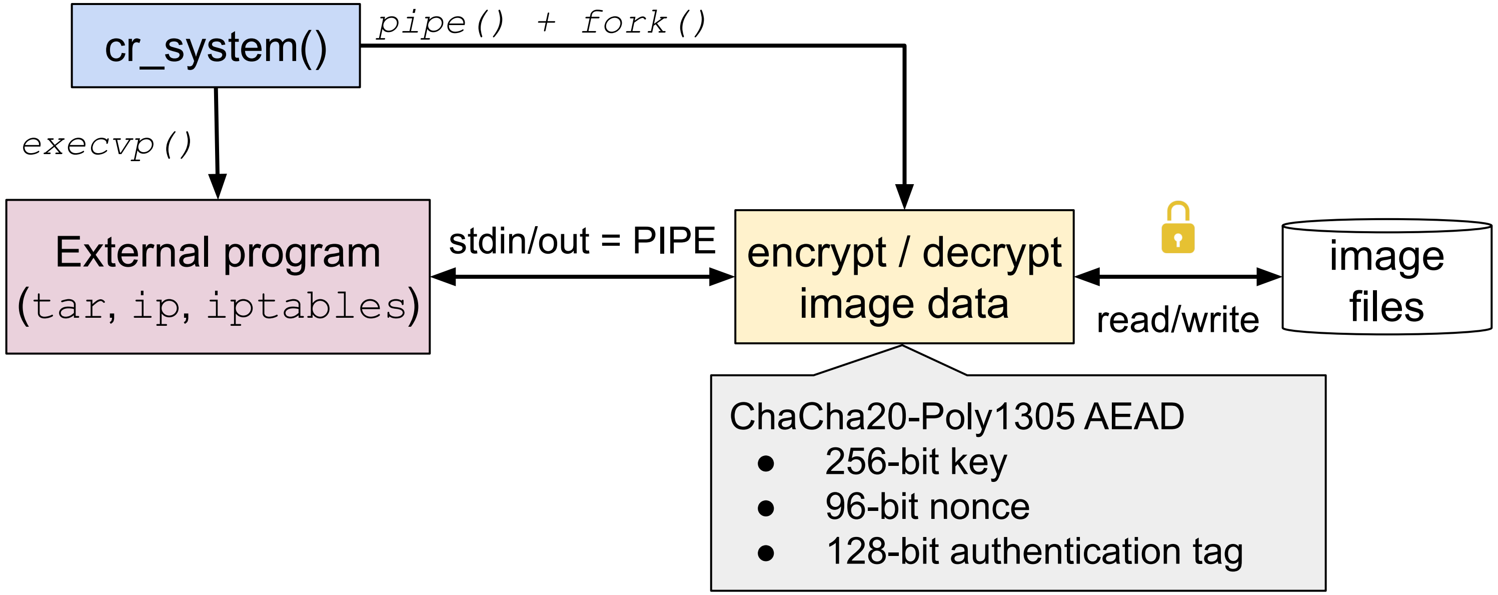
```
syntax = "proto2";
message inventory_entry {
    required uint32 img_version = 1;
    optional bool fdinfo_per_id = 2;
    optional task_kobj_ids_entry root_ids = 3;
    optional bool ns_per_id = 4;
    optional uint32 root_cg_set = 5;
    optional lsmtype lsmtype = 6;
    optional uint64 dump_uptime = 8;
    optional uint32 pre_dump_mode = 9;
    optional bool tcp_close = 10;
    optional uint32 network_lock_method = 11;
}
```



ChaCha20-Poly1305 AEAD

- 256-bit key
- 96-bit nonce
- 128-bit authentication tag

Images in 3rd-party Format



ChaCha20-Poly1305 AEAD

- 256-bit key
- 96-bit nonce
- 128-bit authentication tag

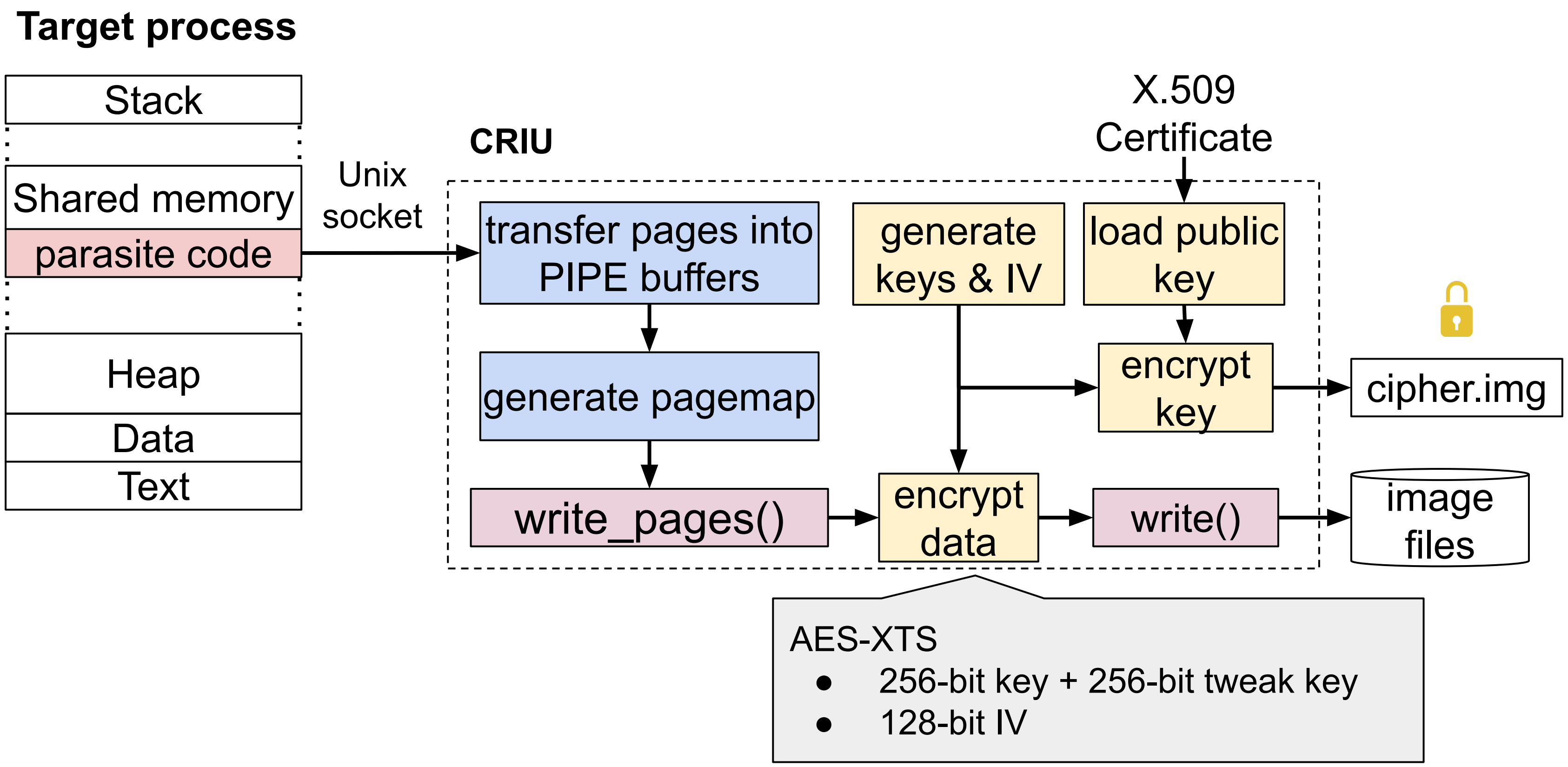
Encryption of Memory Pages

AES-XTS

- XOR-encrypt-XOR (XEX) tweakable block cipher with ciphertext stealing
  - Single IV per checkpoint (reduces storage overhead)
- Memory pages are accessible individually
  - Enables support for iterative checkpointing & memory deduplication
- Hardware acceleration (~7× increased performance [1])

[1] Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz
https://gitlab.com/gnutls/gnutls/-/merge_requests/1244

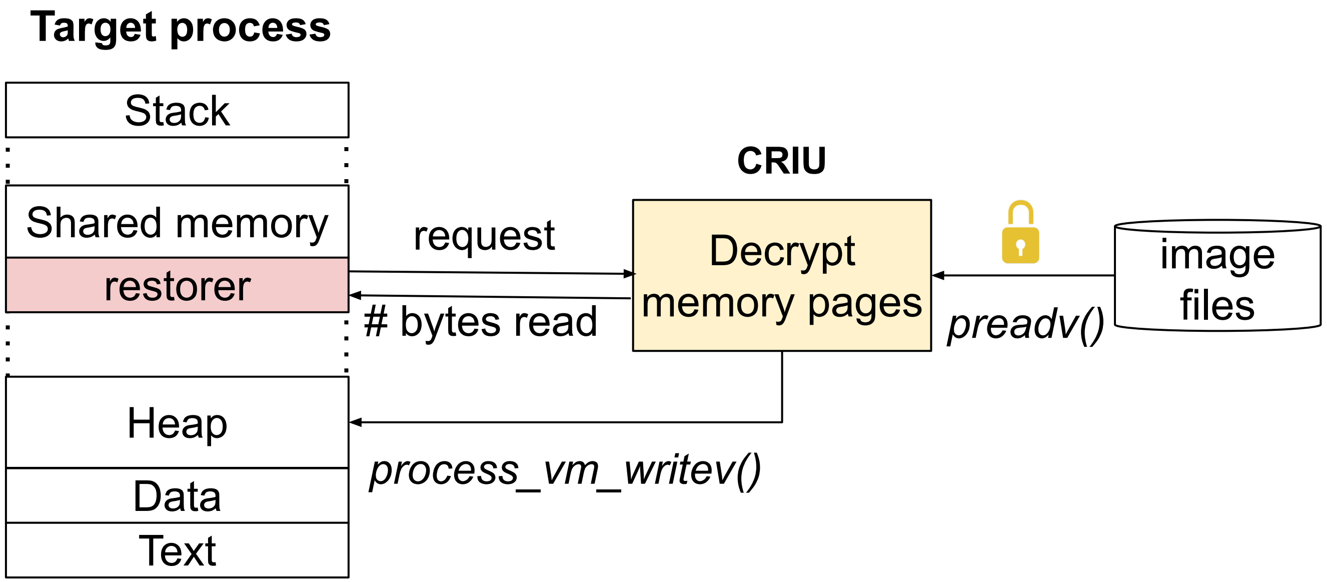
Encryption of Memory Pages



AES-XTS

- 256-bit key + 256-bit tweak key
- 128-bit IV

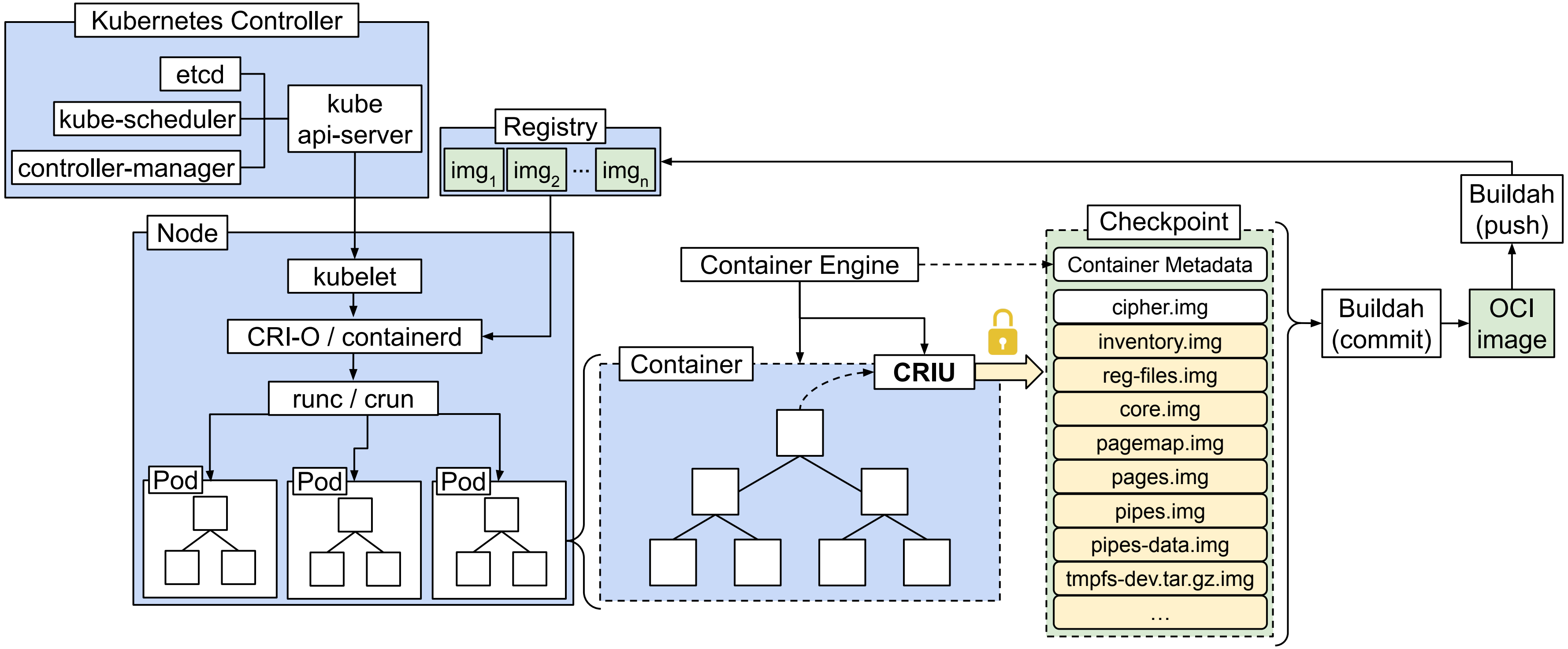
Decryption of Memory Pages



End-to-End Encryption in Kubernetes

Integration with existing container runtimes

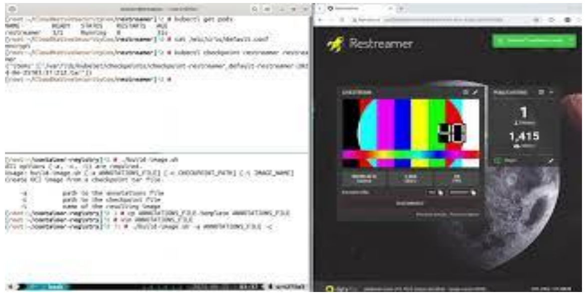
End-to-End Encryption in K8s



Checkpoint Encryption Demo

LLM Inference
(Open-WebUI + Ollama)

In-memory DB
(Redis)



Video Streaming
(Restreamer)

Performance Evaluation

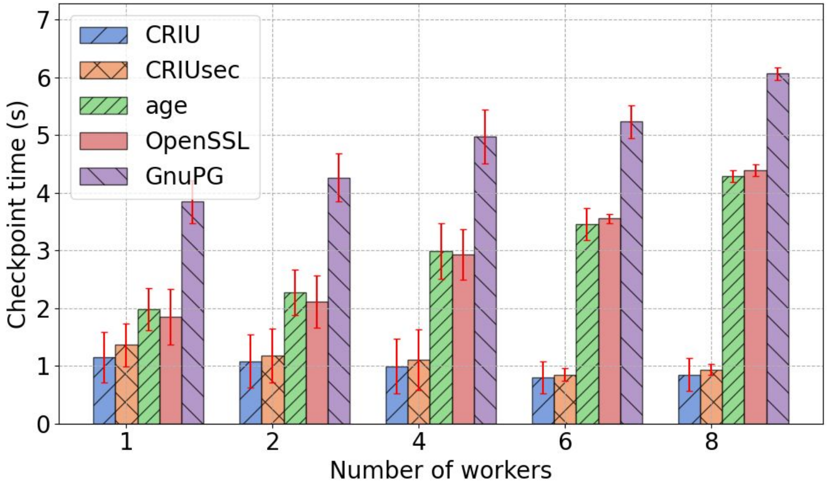
- Workloads
  - Compute-intensive – large number of CRIU images with small size (process tree)
  - Memory-intensive – small number of CRIU images with large size (memory pages)

- Alternative solutions
  - CRIU – Unencrypted checkpoint
  - CRIUsec – CRIU with built-in encryption
  - OpenSSL
  - GnuPG
  - Age

} Action-script called at **post-dump** hook
(https://criu.org/Action_scripts)

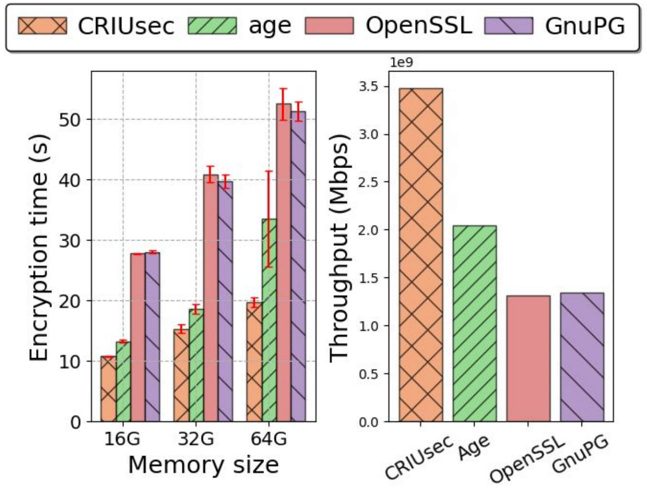
Performance Evaluation

Checkpoint creation time for compute-intensive workloads



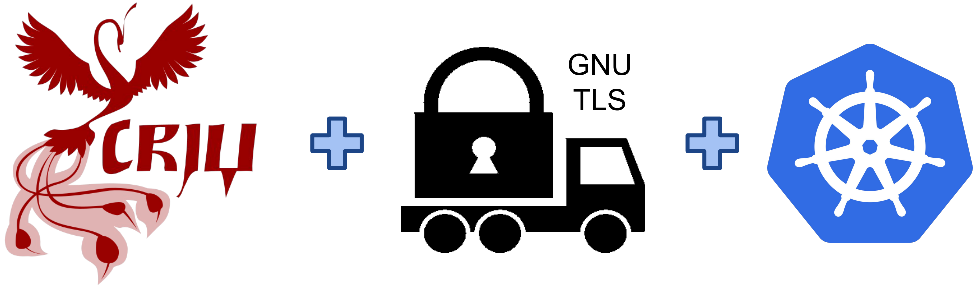
Up to two orders of magnitude faster checkpoint creation

Encryption throughput for memory-intensive workloads



Up to 62% reduced encryption overhead

Summary & Questions?



- Built-in checkpoint encryption support
- Reduced encryption overhead
- Seamless integration with Kubernetes

https://github.com/checkpoint-restore/criu