IEEE SoutheastCon 2024 -
Atlanta, GA
“Engineering the Future”

Application-Level Checkpoint/Restart for Large-Scale Attack and Compliance Graphs

Noah L. Schrick, Peter J. Hawrylak
University of Tulsa

Track#3
Session:
Presentation Date

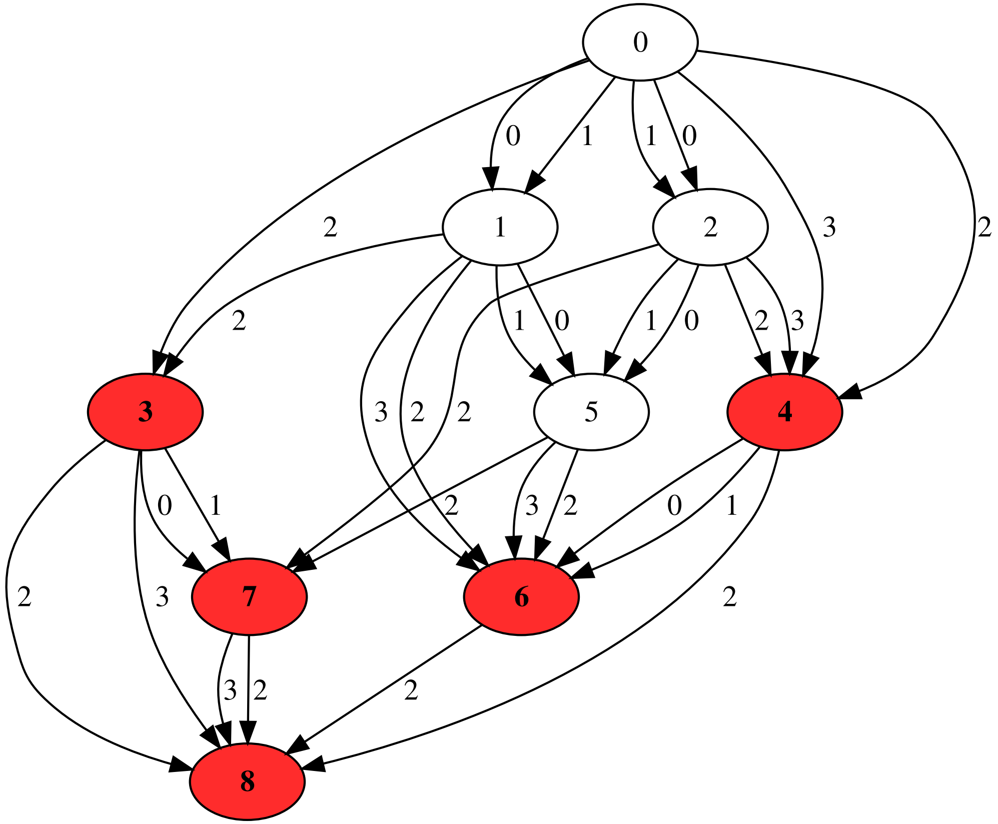
SOUTHEASTCON 2024
Region 3
Engineering the Future

Introduction (1/2)

Overview

- **Attack Graph -**
  - Determine all possible ways systems may be compromised [1]

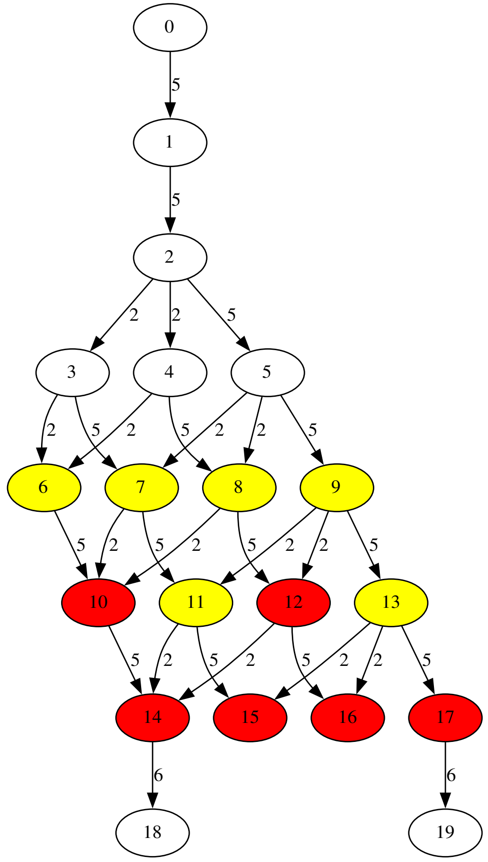
- **Compliance Graph -**
  - Determine all possible ways systems may fall out of compliance [2]



Introduction (2/2)

Terminology/Descriptions

- Nodes
  - States within the graph.
  - Have system information embedded within the object.
    - Example: Windows 10 machine, pfSense firewall vX.Y, 2006 Toyota Corolla
- Edges
  - Transitions within the graph.
  - Events that lead to a change in the system(s) or environment(s).
    - Example: Installing or updating software/hardware, regularly occurring maintenance, spread of malware



Challenges (1/2)

Challenges with Attack and Compliance Graphs

- Scalability (State Space Explosion) [3, 7]
  - The exponential growth of states and edges caused through minimal additions of assets, qualities, or events .
  - Leads to graphs with hundreds of millions of nodes, and billions of edges.

- High Runtime Requirements [3-7]
  - Real-world performance of graph operations does not align with the theoretical assumption.
  - Scalability – large graphs take exceedingly long to generate.
    - Example: Installing or updating software/hardware, regularly occurring maintenance, spread of malware

Challenges (2/2)

Implications

- Graphs and graph operations cannot be contained within non-volatile memory (RAM).
  - Out-of-memory killers will terminate the generation process.

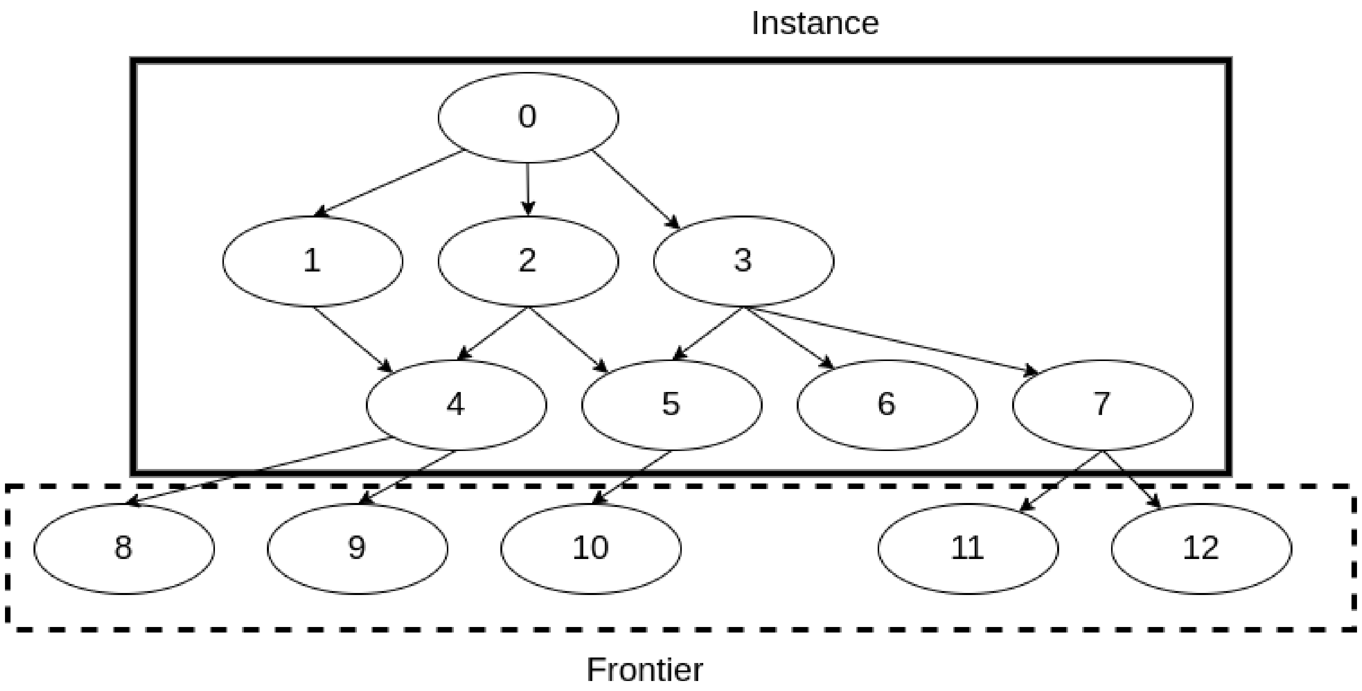
- Outages, HPC cycle exhaustion, or other interruption forces a complete re-generation of the graphs.
  - Can result in a loss of weeks' worth of processing.

Memory Constraint

Two Primary Pain Points

1) The queue of unexplored nodes
- "Frontier"
- Caused by the Breadth-First Search generation approach

2) The graph object
- "Instance"
- All explored nodes (and their embedded information), edges, flags, or auxiliary graph labels or features



Related Works

Specific to Attack and Compliance Graphs

- Efficient storage techniques [13, 14].
- Logic-based generation [15].
- Alternate information representation schemes [16, 17].
- Sampling [18].
- Parallelization [19].

Checkpoint/Restart (C/R)

Introduction

- A technique that saves the state of a program mid-execution, and allows for a restart from a saved state.

- Three categories [8, 9]:
  - System-level
    - Requires compatibility with the operating system, and any application libraries (e.g., MPI).
    - Large in scope: can restore process IDs, checkpoint shell scripts, sockets, threads, file processing.
  - User-level
    - Large, application-independent checkpoints that are linked through libraries.
  - Application-level
    - Built into an application’s source code.
    - Goal: only handle the necessary information.

Goal of This Work

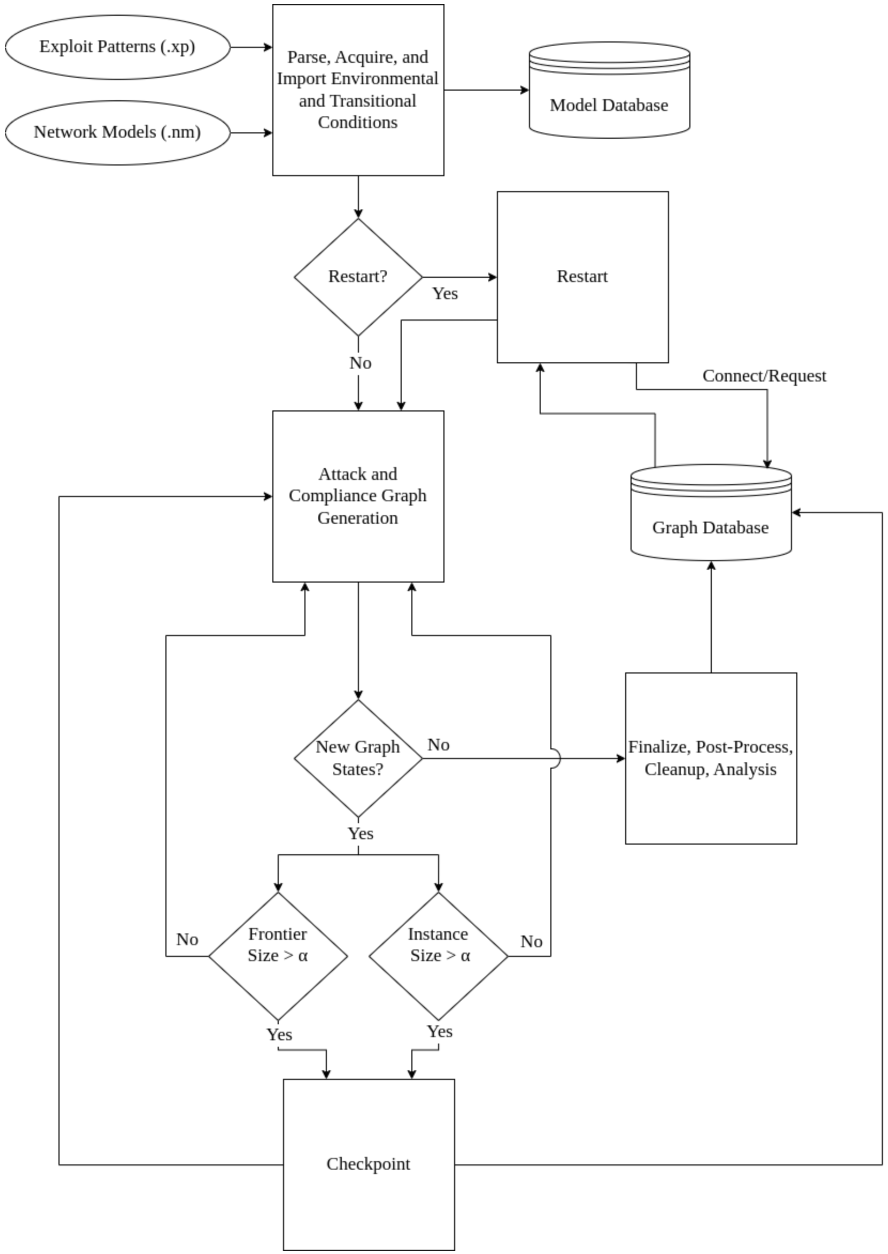
Implement Application-level C/R

- Minimal checkpoints for fast, efficient checkpoint and restart procedures.
  - C/R will also be portable, and independent of external libraries or operating systems.

- Benefits are twofold:
  - Provides a form of fault-tolerance in the event of interruption.
  - Provides a means of memory relief by dumping excess, no longer relevant graph instances during checkpoint intervals.

Overview

Generation Process



Checkpointing

Implementation Details

- Dynamic storing
  - Based on available memory.
  - New PostgreSQL table for the frontier.
    - Graph instance tables already existed.
    - Maintain proper ordering of FIFO queue.

- Alpha (α) Parameter
  - (0, 1.0)
    - Checkpoint when a graph instance consumes a percentage of memory relative to allocated.
  - [1.0, n)
    - Checkpoint when the graph instance has *n* nodes.
  - Ideally, user will pass in memory requests to a job scheduler (e.g., Slurm), and RAGE.

Checkpointing

Implementation Details

- Processed during the OpenMP critical section, but can be passed to the dedicated MPI database node to allow for a continuation in generation.
- Memory buffer is reserved for building SQL queries.
- No file system dependencies required.
- Abstracted from PostgreSQL specific implementations, so PostgreSQL redundancies can be added to the cluster with no cost to application runtime.

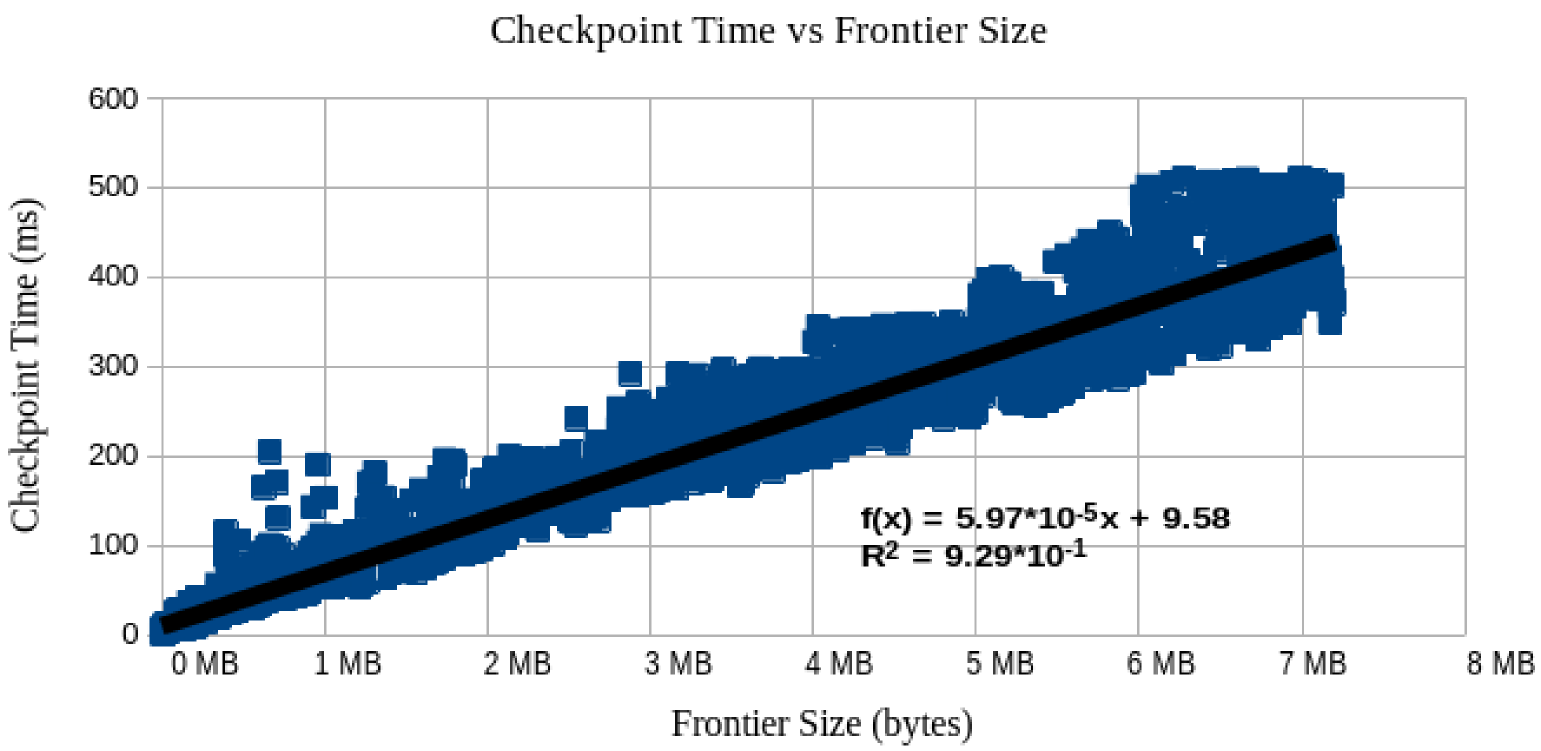
Restarting

Implementation Details

- New restart parameter.
- Pulls the frontier.
- Recovers node ID positioning.
- Graph instance was already explored: no need to recover the known instance.

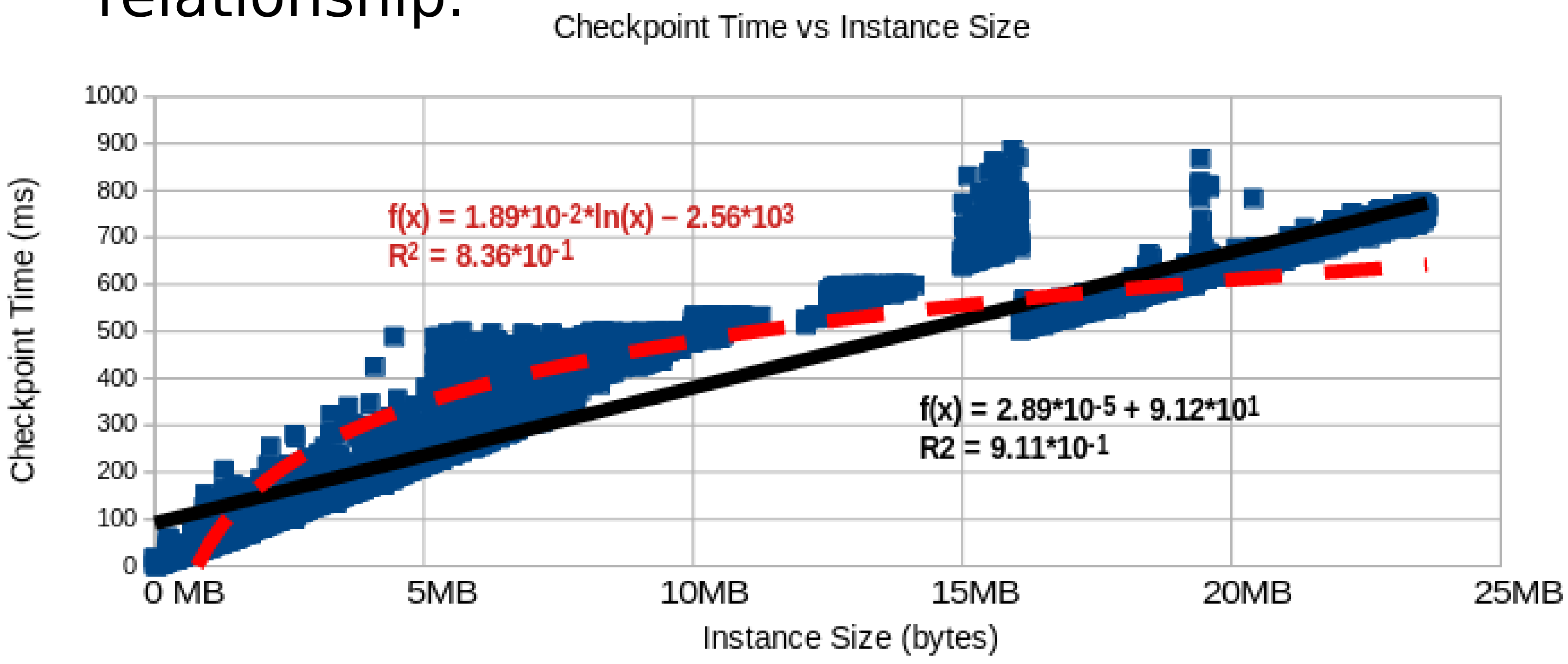
Results - Checkpointing

- Linear relationship between checkpoint time and frontier size.



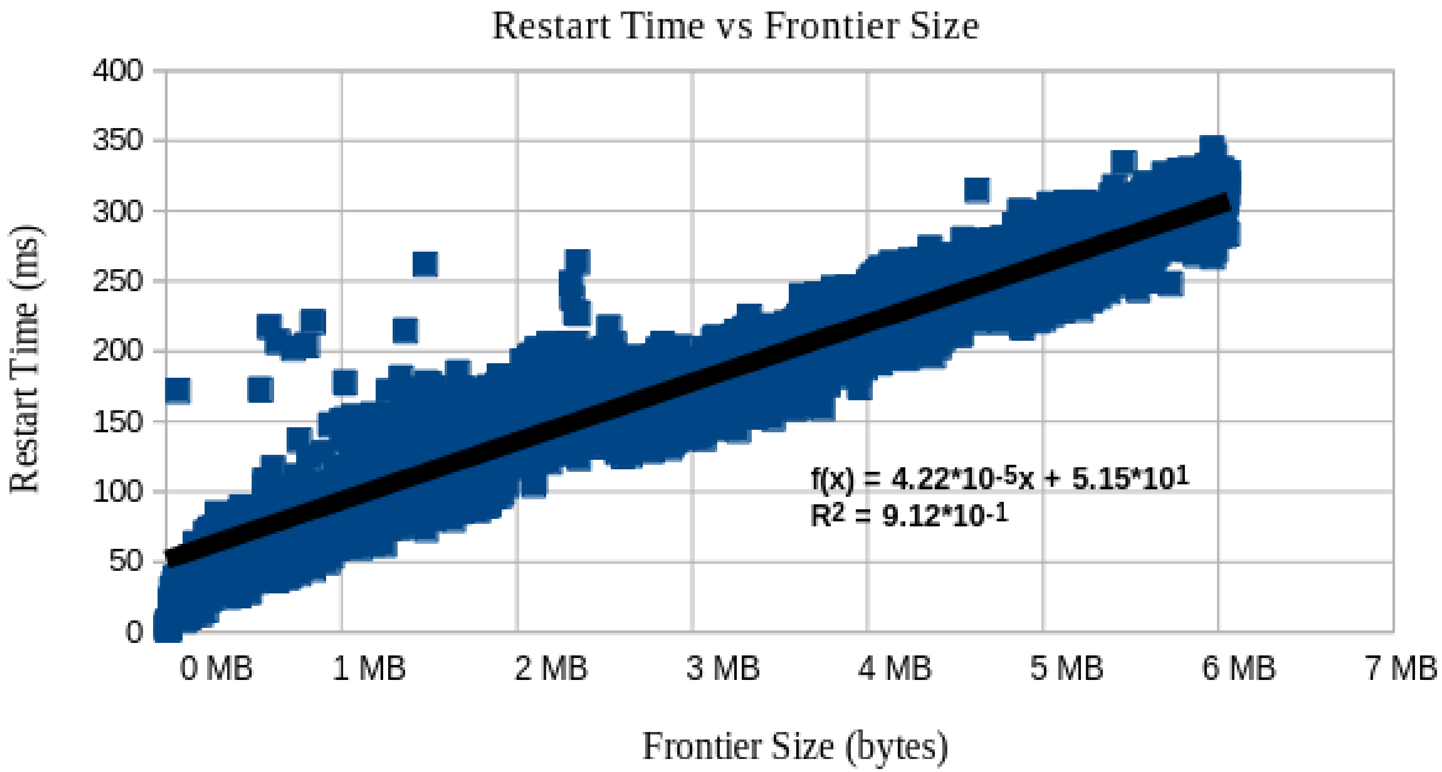
Results - Checkpointing

- Bound by a logarithmic relationship between checkpoint time and instance size when the instance is smaller.
- As the instance grows, checkpoint time better fits to a linear relationship.



Results - Restarting

- Linear relationship between restart time and frontier size.



Conclusions

Implemented Application-level CR that:

- Successfully alleviates “unnecessary” memory pressure for large graphs.
- Minimally checkpoints graph objects and an unexplored queue of nodes without consuming an excessive amount of runtime.
- Restarts to a known state in the event of an interruption without consuming an excessive amount of runtime.

Future Work

Should Investigate:

- Comparison to known C/R libraries:
  - Scalable Checkpoint/Restart (SCR) [10].
  - Distributed MultiThreaded Checkpointing (DMTCP) [11].
  - Berkely Lab Checkpoint/Restart [12].
- Optimize database queries and/or database configurations.
- Filesystem C/R.
- Optimizing checkpointing interval [21].

Thank You!

IEEE 140 YEARS 1884 - 2024

IEEE Advancing Technology for Humanity