From Dynamic Loading to Extensible Transformation: An Infrastructure for Dynamic Library Transformation

Yuxin Ren, Kang Zhou, Jianhai Luan, Yunfeng Ye,
Shiyuan Hu, Xu Wu, Wenqin Zheng, Wenfeng Zhang, Xinwei Hu
Huawei Technologies

Background： dynamic library

- Modularity

  complex software can be developed, delivered, and distributed as a collection of libraries, instead of a single binary.

Background： dynamic library

- Modularity

  complex software can be developed, delivered, and distributed as a collection of libraries, instead of a single binary.

- Maintainability

  dynamic library can be updated or patched individually without modifying or re-compiling the entire application.

Background： dynamic library

- Modularity

  complex software can be developed, delivered, and distributed as a collection of libraries, instead of a single binary.

- Maintainability

  dynamic library can be updated or patched individually without modifying or re-compiling the entire application.

- Sharability

  dynamic library can be shared among multiple applications, thus avoid duplication in disk or memory

Background： dynamic library

- Modularity

  complex software can be developed, delivered, and distributed as a collection of libraries, instead of a single binary.

- Maintainability

  dynamic library can be updated or patched individually without modifying or re-compiling the entire application.

- Sharability

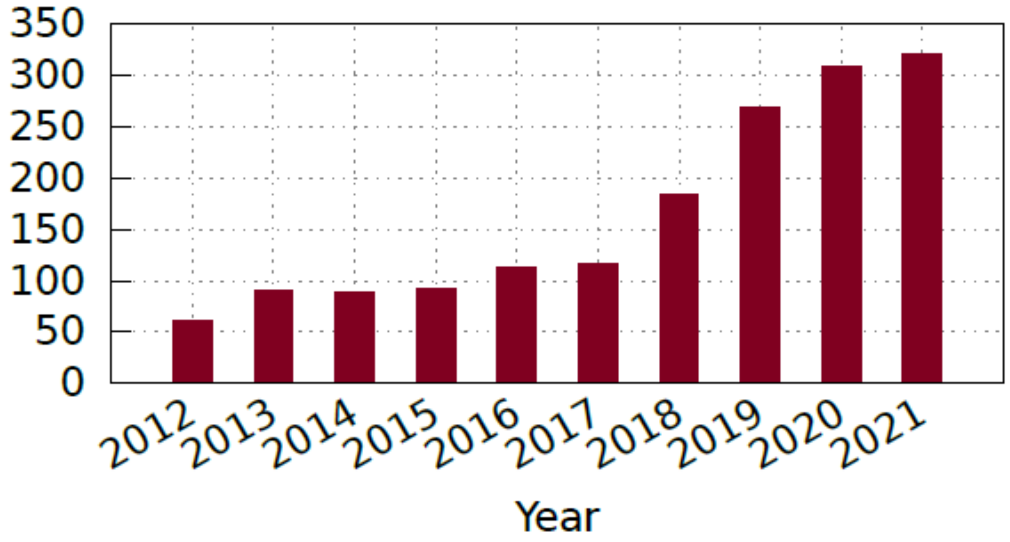
  dynamic library can be shared among multiple applications, thus avoid duplication in disk or memory

- Open source friendly

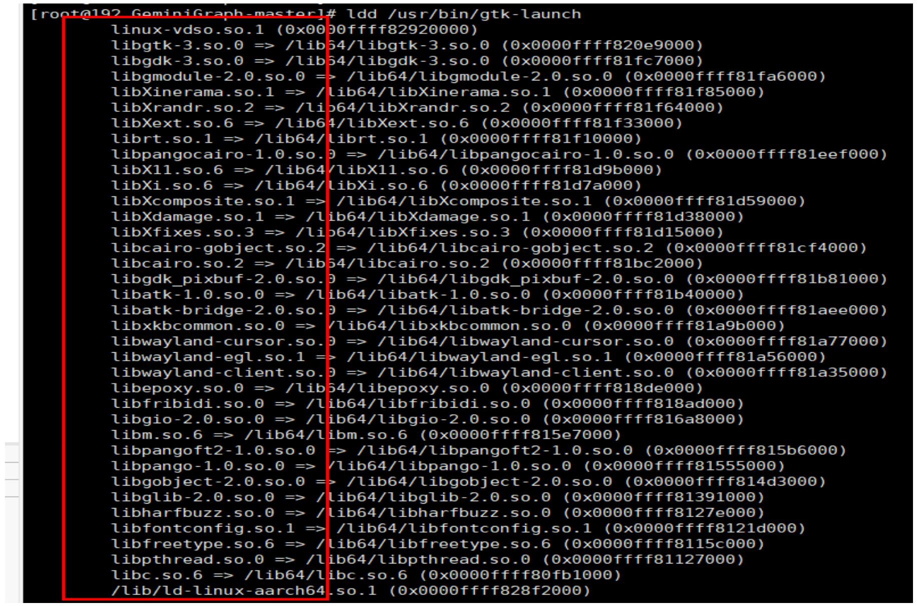
  license contamination： open source license requires all statically linked code should also be open-sourced

Background： dynamic library

- More and more dynamic libraries are shipped by vendors
- More and more dynamic libraries are used by applications



The number of dynamic libraries included in the CUDA Toolkit over the past decade



Applications can rely on from tens to hundreds dynamic libraries

Background： performance overhead

Memory management： each library is individually mapped into the process's address space. Invocation between libraries touch different pages, incurring TLB miss

	TLB miss	IPC	99th percentile latency (cycle)	Execution time (s)
glibc	1,231,950	1.96	318	6.01
iFed	117,782	2.43	232	4.86

A micro-benchmark that simply invokes 100 dynamic libraries, and each library contains only one function accessing memory.
Performance comparison between glibc and iFed on x86 machine.

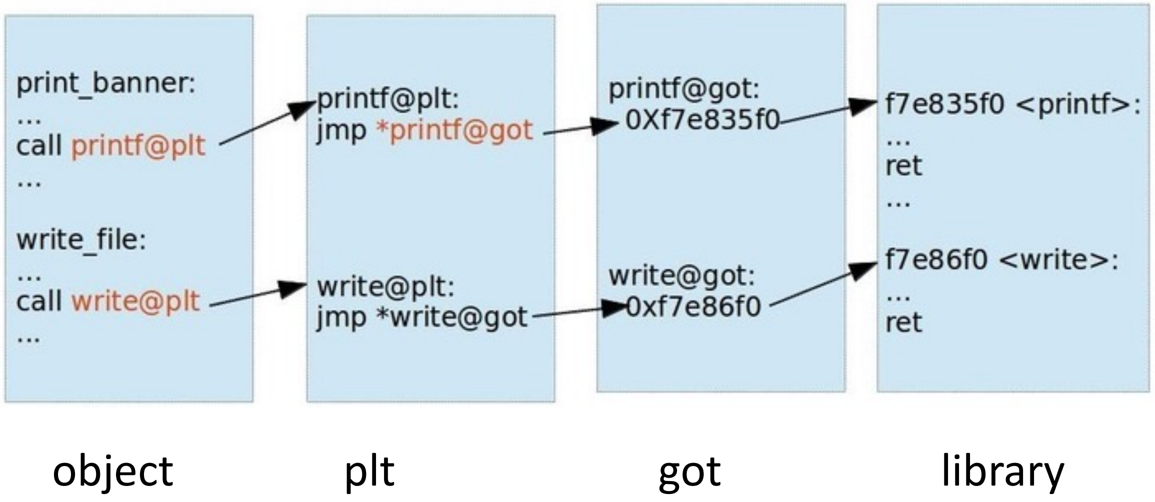
Background： performance overhead

Memory management： each library is individually mapped into the process's address space. Invocation between libraries touch different pages, incurring TLB miss

	TLB miss	IPC	99th percentile latency (cycle)	Execution time (s)
glibc	1,231,950	1.96	318	6.01
iFed	117,782	2.43	232	4.86

A micro-benchmark that simply invokes 100 dynamic libraries, and each library contains only one function accessing memory.
Performance comparison between glibc and iFed on x86 machine.

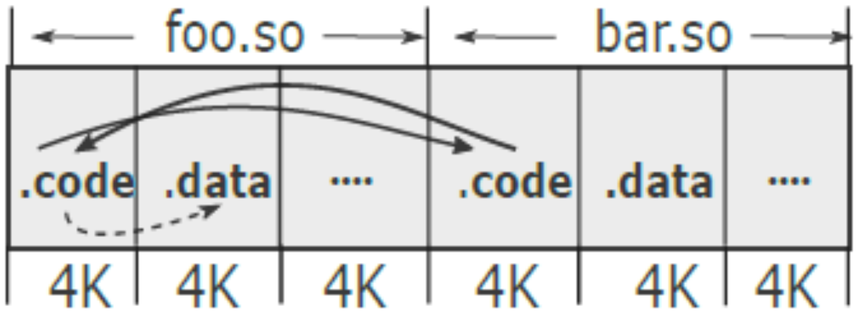
Relocation： more memory access and executed instructions incur extra branch miss, cache miss, etc…



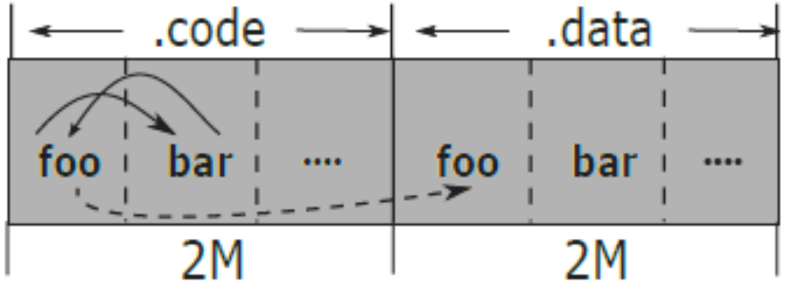
simplified execution flow of relocation

Optimizations: Dynamic Library Concatenation

- Collect the same sections, such .code, from all dynamic libraries and concatenate them one by one to form a big section.
- This combined section is large enough to fit in hugepages



Different sections in different libraries use small page



Same sections in different libraries are combined and use hugepage

Optimizations: Dynamic Library Concatenation

Trade off

- Reduced address space layout randomization

  Mitigations：

  (1) concatenate libraries in random order.

  (2) non-continuous Hugepages.

  (3) leverage other code randomization techniques at load time

- Reduced library sharing

  Mitigations

  (1) Only apply to performance critical applications

  (2) Multiple forked instances can still share combined libraries

  (3) Sharing part of a hugepage

Optimizations: Relocation Branch Elimination

- Rewrite the call instructions to replace their target address with the address of library functions, instead of using indirect jump
- Eliminate the extra memory access and branch instruction, achieve similar effect as static linking

```
.text
.foo1:
    call    bar@plt
.foo2:
    call    bar@plt
.plt
bar@plt
    jump  *(bar@got)
.got
    bar    bar@bar.so
```

```
.text
.foo1:
    call    bar@bar.so
.foo2:
    call    bar@bar.so
```

Optimizations: Relocation Branch Elimination

Trade off

- Increased loading time

  Mitigations：

  (1) Little impact on long-running services, such as web server and database

  (2) Apply in-memory caching technology to load the transformed image

- Increased binary size

  Mitigations

  (1) Download on-demand from remote storage

  (2) Compresses binary

Optimazations: more

There is a large body of research focusing on load time technology.

ASPLOS'20

Session 2B: Dynamic compilation — Who moved my cheese?

Egalito: Layout-Agnostic Binary Recompilation

David Williams-King
dwk@cs.columbia.edu
Columbia University

Hidenori Kobayashi
hidenori.kobayashi@gmail.com
Canon Inc.[†]

Kent Williams-King
kwk@brown.edu
Brown University

Frank Spano

Yu Jian Wu
Columbia University

Binary Stirring: Self-randomizing Instruction Addresses of Legacy x86 Binary Code

CCS'12

Richard Wartell, Vishwath Mohan, Kevin W. Hamlen, Zhiqiang Lin
Department of Computer Science, The University of Texas at Dallas
800 W. Campbell Rd, Richardson, TX, 75252
{richard.wartell, vishwath.mohan, hamlen, zhiqiang.lin}@utdallas.edu

Developer and User-Transparent Compiler Optimization for Interactive Applications

Paschalis Mpeis
University of Edinburgh
Edinburgh, UK
p.mpeis@ed.ac.uk

Pavlos Petoumenos
University of Manchester
Manchester, UK
pavlos.petoumenos@manchester.ac.u

Kim Hazelwood
Facebook AI Research
Menlo Park, CA, USA
kimhazelwood@fb.com

Hugh Leather
Facebook AI Research
Menlo Park, CA, USA
hleather@fb.com

PLDI'21

ASLR-Guard: Stopping Address Space Leakage for Code Reuse Attacks

CCS'15

youngyoung Lee, Simon P. Chung,
, and Wenke Lee
e, Georgia Institute of Technology

PLDI'20

BlankIt Library Debloating

Getting What You Want Instead of Cutting What You Don't

Chris Porter*
Georgia Institute of Technology
Atlanta, GA, USA
porter@gatech.edu

Girish Mururu*
Georgia Institute of Technology
Atlanta, GA, USA
girishmururu@gatech.edu

Prithayan Barua
Georgia Institute of Technology
Atlanta, GA, USA
prithayan@gatech.edu

Santosh Pande
Georgia Institute of Technology
Atlanta, GA, USA
santosh.pande@cc.gatech.edu

Debloating Software through Piece-Wise Compilation and Loading

Anh Quach
Binghamton University
aquach1@binghamton.edu

Aravind Prakash
Binghamton University
aprakash@binghamton.edu

Lok Yan
Air Force Research Laboratory
lok.yan@us.af.mil

USENIX Security'18

Problems

- hard to develop

    Require knowledge about the whole loader, introduce ad-hoc and intrusive modifications.

Problems

- hard to develop

  Require knowledge about the whole loader, introduce ad-hoc and intrusive modifications.

- hard to maintain

  Customized patch with application specific optimizations are hard to be accepted by upstream

  .

Problems

- hard to develop

  Require knowledge about the whole loader, introduce ad-hoc and intrusive modifications.

- hard to maintain

  Customized patch with application specific optimizations are hard to be accepted by upstream

  .

- hard to distribute

  Multiple distributions to server different productions with various features

Problems

- hard to develop

  Require knowledge about the whole loader, introduce ad-hoc and intrusive modifications.

- hard to maintain

  Customized patch with application specific optimizations are hard to be accepted by upstream

  .

- hard to distribute

  Multiple distributions to server different productions with various features

- hard to use

  Cannot combine different features for different application scenario

Problems

- hard to develop

  Require knowledge about the whole loader, introduce ad-hoc and intrusive modifications.

- hard to maint

  Customized patch upstream

  .

- hard to distrib

  Multiple distribut

- hard to use

  Cannot combine different features for different application scenario

Root cause:
Monolithic design with
no interface to allow extensions

New loader: goals

- Extensibility and Modularity

    Various functionality should be organized in a loosely-coupled way instead of a monolithic implementation

New loader: goals

- Extensibility and Modularity

  Various functionality should be organized in a loosely-coupled way instead of a monolithic implementation

- Flexibility and Customizability

  flexibly configured for different trade-off on per-application, customer, or even per-run basis

New loader: goals

- Extensibility and Modularity

  Various functionality should be organized in a loosely-coupled way instead of a monolithic implementation

- Flexibility and Customizability

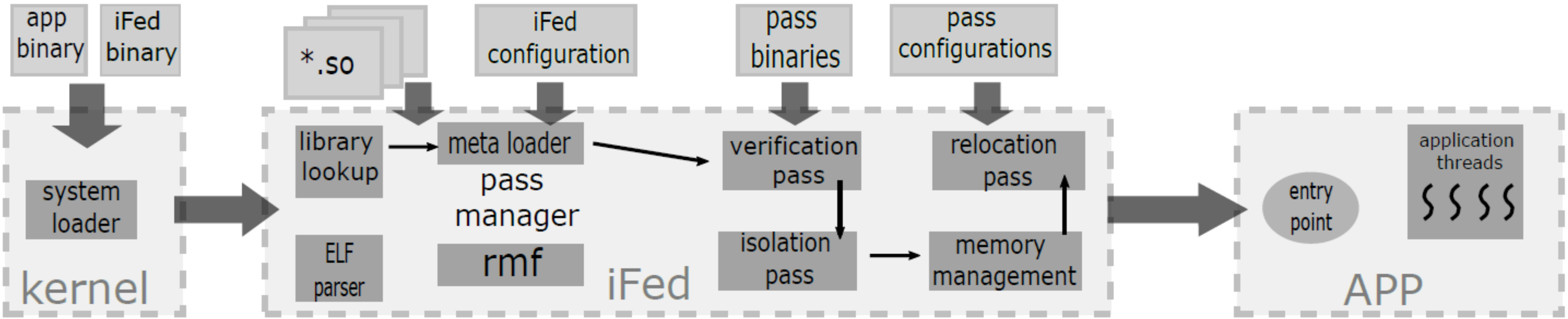
  flexibly configured for different trade-off on per-application, customer, or even per-run basis

- Compatibility and Transparency

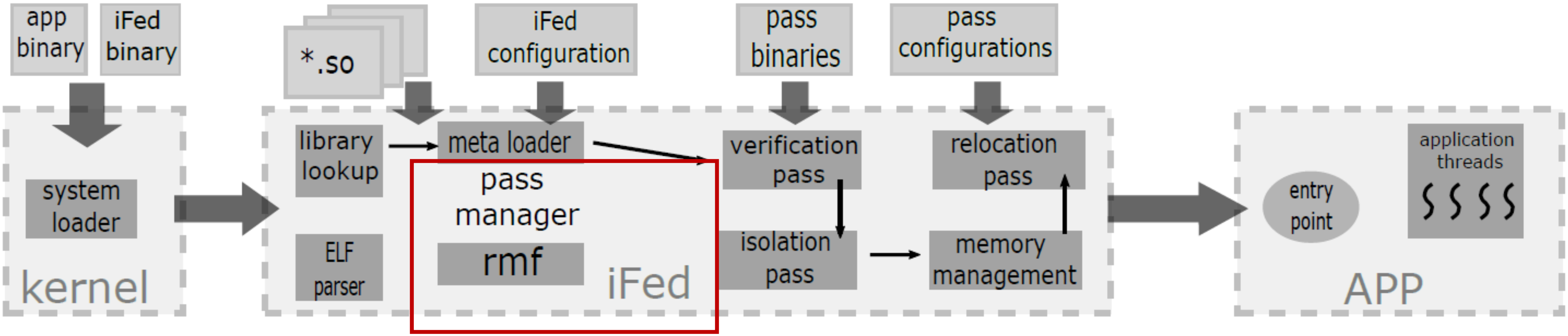
  Compatible with the existing loader interface and transparent to application

New loader: iFed overview

iFed (infrastructure for flexible and extensible dynamic library transformation)

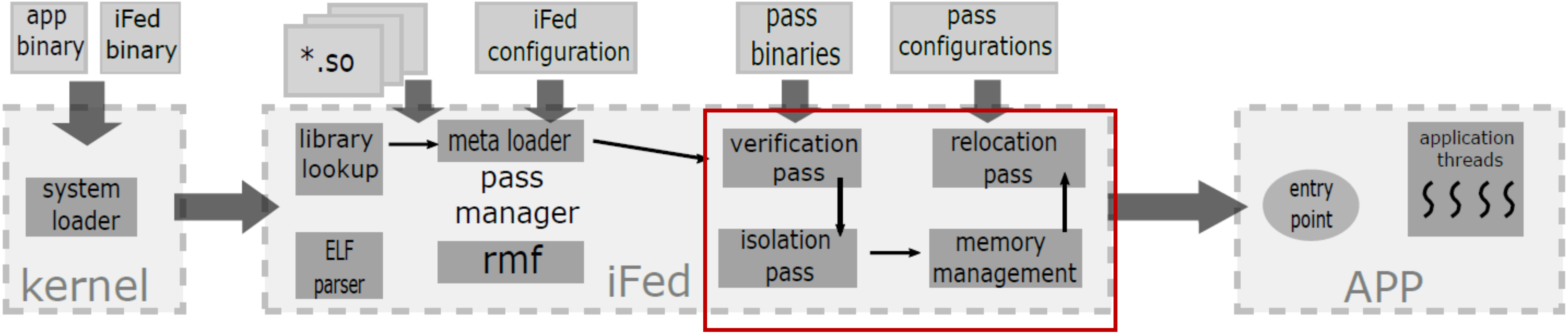


New loader: key technique



- Runnable in-memory format
  - ELF is for dense storage on disk, a in-memory counterpart is missing
  - Abstract around common information and states, such as relocations and symbols
  - Collect all information from all libraries for global optimization
  - Expose unified interface to upper library transformation

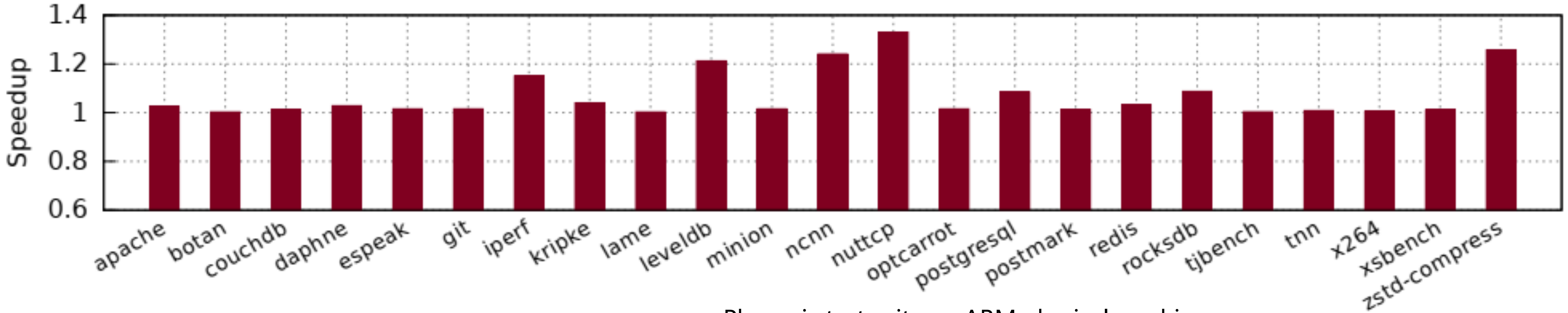
New loader: key technique



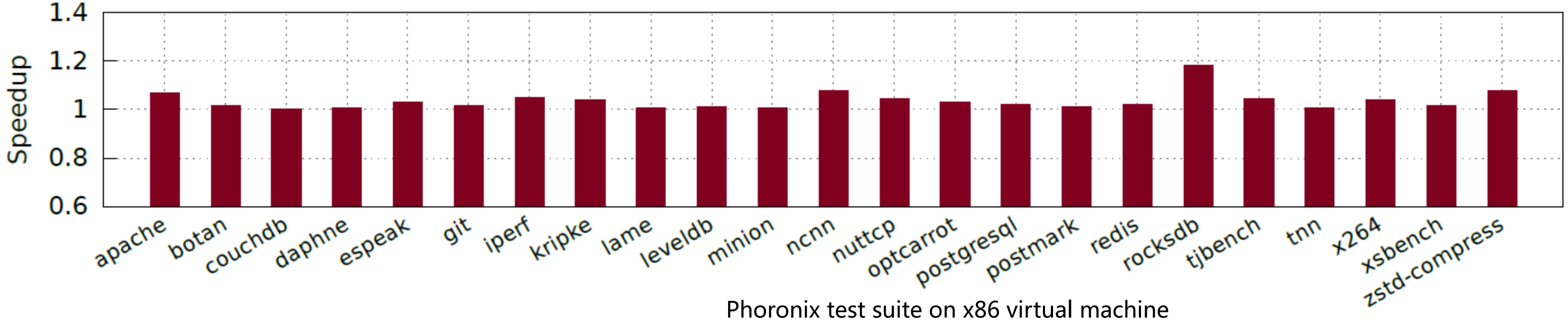
- Pass-based optimization framework
  - Library transformation is implemented as separated pass
  - Multiple passes form a pipeline
  - Passes interact via RiMF

New loader: evaluation

We evaluate iFed with a large range of application



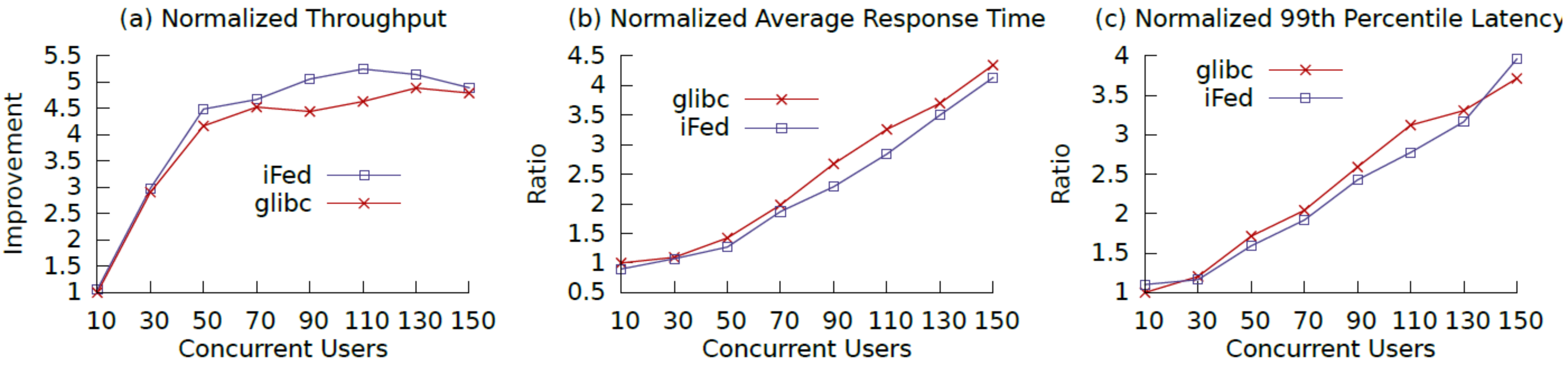
Phoronix test suite on ARM physical machine



Phoronix test suite on x86 virtual machine

New loader: evaluation

evaluate iFed on multiple performance dimensions with a dynamic social website



Dynamic web serving performance

New loader: open question and discussion

- Loader Functionality
  - Memory management
  - Isolation
  - Security enhancement
  - Binary rewriting and execution control

- Other linker and loader architecture

- License： Is it reasonable to rely on the type of linking?

Conclusion

- A pass-based infrastructure for extensible, flexible, and modular transformation on dynamic library
- Two performance optimization passes
  - Dynamic Library Concatenation
  - Relocation Branch Elimination

Open source communities

OpenHarmony,OpenEuler, OpenGauss, MindSpore

Most active 5.10 employers

By changesets			By lines changed		
Huawei	1434	8.9%	Intel	96976	12.6%
Intel	1297	8.0%	Huawei	41049	5.3%

Most active 5.8 employers

By changesets			By lines changed		
Intel	1939	11.9%	Huawei	293365	27.8%
Huawei	1399	8.6%	Intel	93213	8.8%

https://lwn.net/Articles/839772/

Huawei is one of the top contributor in Linux community

OpenHarmony

https://www.openharmony.cn/

OpenHarmony is an open-source project incubated and operated by the OpenAtom Foundation. It is an open-source operating system with a framework and platform applicable to smart devices in all scenarios of a fully-connected world. It aims to promote the development of the Internet of Everything (IoE).

openEuler

https://openeuler.org/en

6415 Contributors

91 SIGs

As an open community, openEuler works with global developers to build an open, diverse, and architecture-inclusive software ecosystem that supports multiple processor architectures and covers a full range of digital facilities. openEuler is committed to supercharging enterprise digital infrastructure and boosting the application ecosystem.

openGauss

https://opengauss.org/en/

1.9k Developer

22.2k Users

openGauss is an open source relational database management system that is released with the Mulan PSL v2. with the kernel built on Huawei's years of experience in the database field and continuously provides competitive features tailored to enterprise-grade scenarios.

MindSpore

https://www.mindspore.cn/en

626.5K+ Total Downloads

18.2K+ Total Starred

MindSpore is a deep learning framework in all scenarios, aiming to achieve easy development, efficient execution, and all-scenario coverage.