Efficient Implementation of the bare-metal Hypervisor MetalSVM for the SCC

Public Release of MetalSVM 0.1

Pablo Reble, Jacek Galowicz, Stefan Lankes and Thomas Bemmerl

MARC Symposium, Toulouse

July 20, 2012

CHAIR FOR OPERATING SYSTEMS
Univ.-Prof. Dr. habil. Thomas Bemmerl

RWTH AACHEN UNIVERSITY

Introduction

- What is MetalSVM?
- Our Progress with MetalSVM
- Main Features
- Performance
- How to Get and Use MetalSVM

What is MetalSVM?

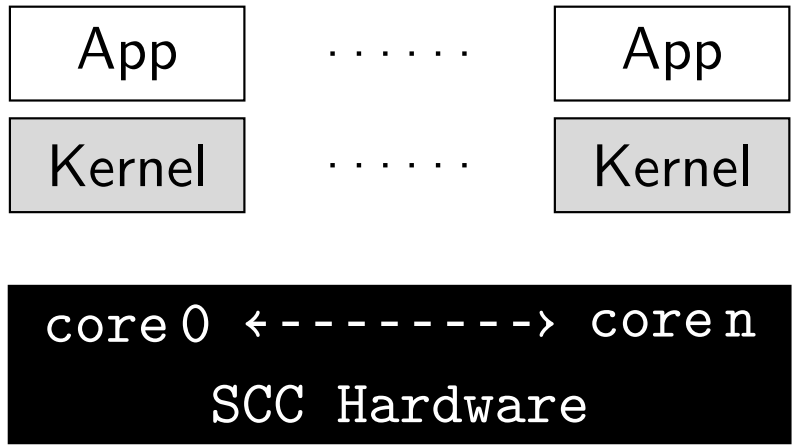
- a **minimal** operating system for the SCC
- spin-off from *eduOS* - a kernel developed at RWTH Aachen for educational purposes
- monolithic, *MULTIBOOT*-compliant 32 bit **lightweight** kernel
- **unix-like** programming interface
- sophisticated exploitation of SCC hardware
- merged with our *iRCCE* communication lib

What is MetalSVM?

- First public release
  → Version 0.1
- Free Software
  → Apache License 2.0
- Is probably a great springboard for your bare-metal programming
- great mascot
  → Mike the Ostrich

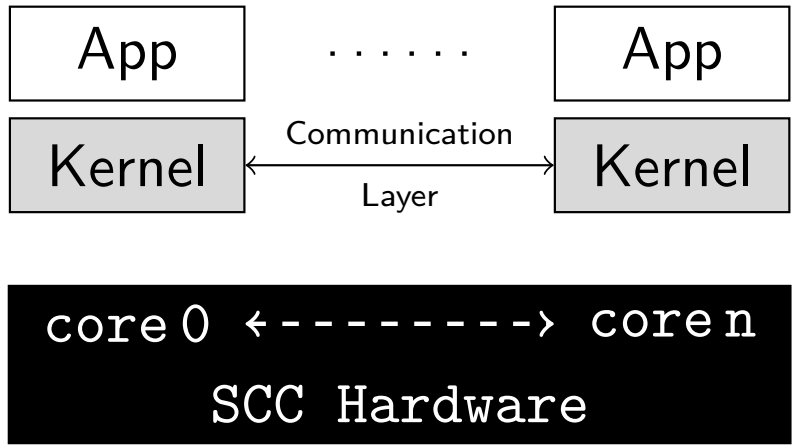
Our Progress with MetalSVM

- run apps with small efficient kernel ✓
- Integrate iRCCE for **fast** inter-core communication ✓
- Establish SVM between multiple cores ✓
- Paravirtualize Linux: One instance per core ✓
- Run one guest on multiple cores [in progress]



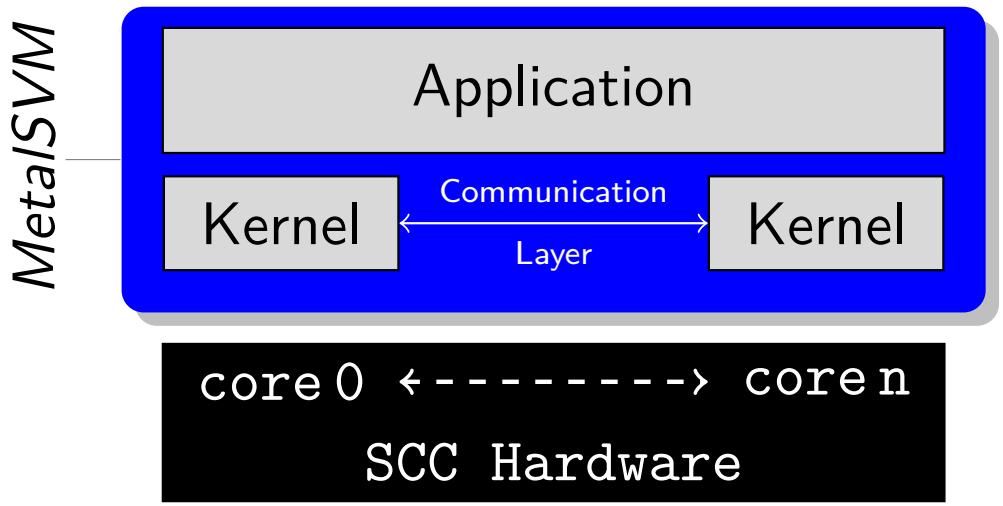
Our Progress with MetalSVM

- run apps with small efficient kernel ✓
- Integrate iRCCE for **fast** inter-core communication ✓
- Establish SVM between multiple cores ✓
- Paravirtualize Linux: One instance per core ✓
- Run one guest on multiple cores [in progress]



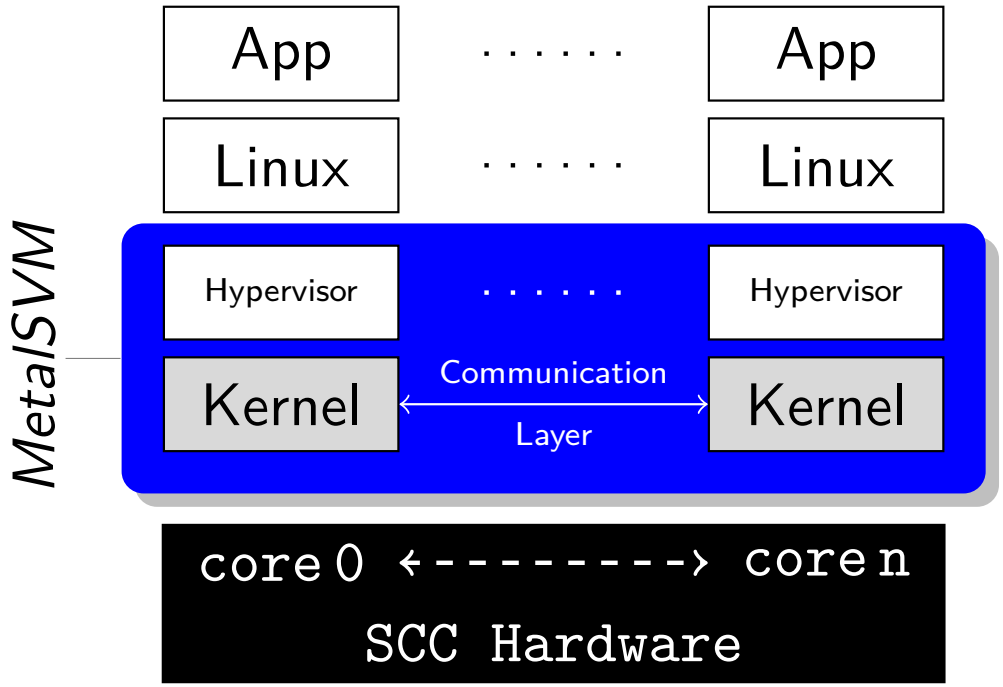
Our Progress with MetalSVM

- run apps with small efficient kernel ✓
- Integrate iRCCE for **fast** inter-core communication ✓
- Establish SVM between multiple cores ✓
- Paravirtualize Linux: One instance per core ✓
- Run one guest on multiple cores [in progress]



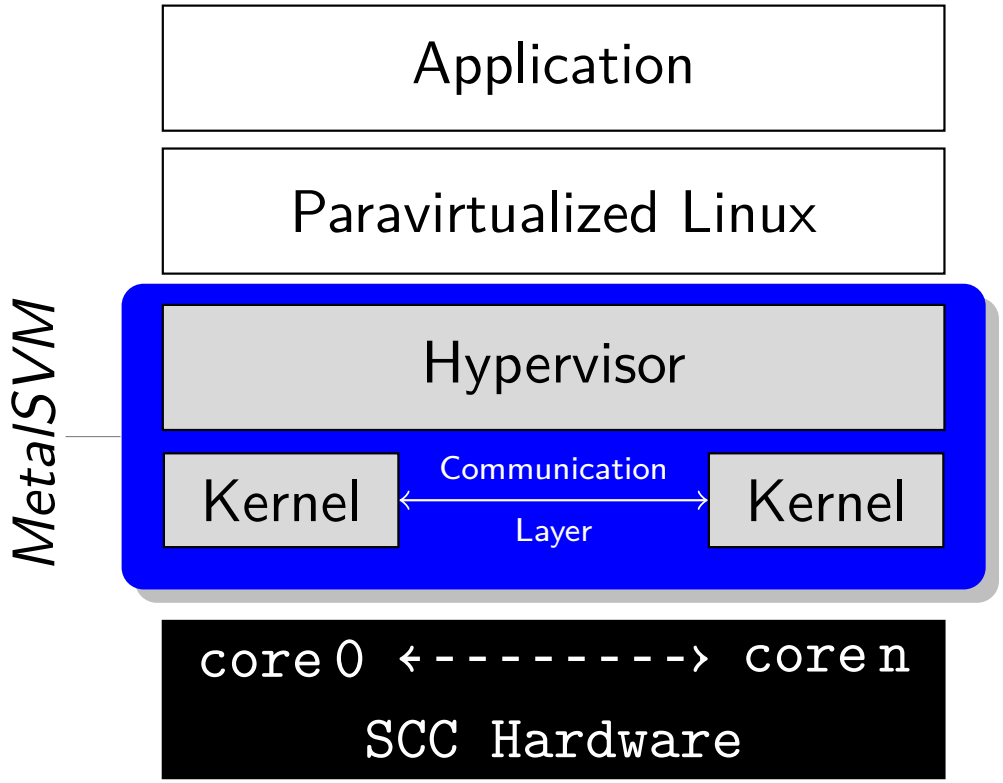
Our Progress with MetalSVM

- run apps with small efficient kernel ✓
- Integrate iRCCE for **fast** inter-core communication ✓
- Establish SVM between multiple cores ✓
- Paravirtualize Linux: One instance per core ✓
- Run one guest on multiple cores [in progress]



Our Progress with MetalSVM

- run apps with small efficient kernel ✓
- Integrate iRCCE for **fast** inter-core communication ✓
- Establish SVM between multiple cores ✓
- Paravirtualize Linux: One instance per core ✓
- Run one guest on multiple cores [in progress]



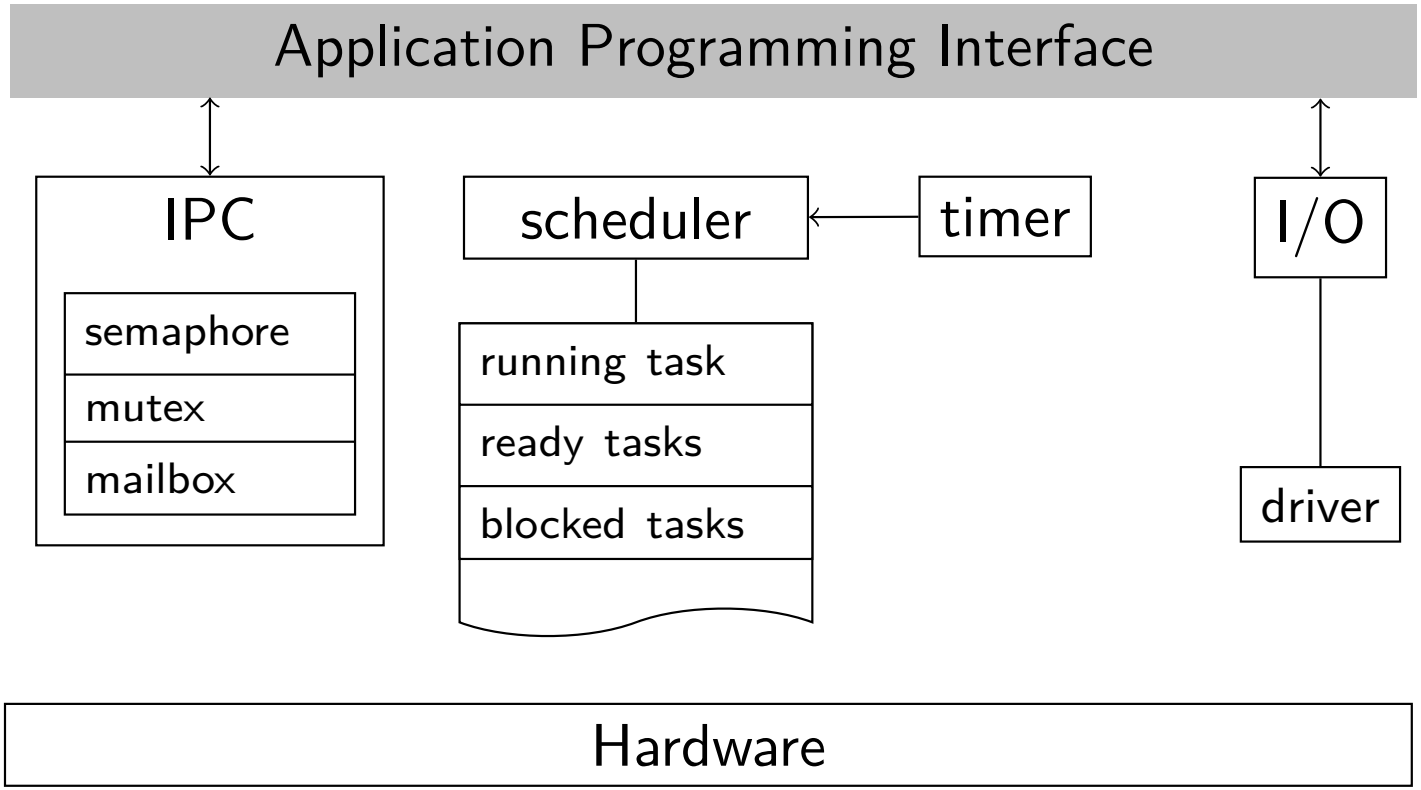
Main Features

- running apps on MetalSVM is like running them **bare-metal** $\rightarrow$ but not without the comfort of an OS!
- fast familiarization due to imitated aspects of Unix-like systems
- *iRCCE* included: **fast** communication
  - core $\leftrightarrow$ core
  - functionality of *RCCE* + **non-blocking** message-passing
  - **event-driven** *Mailbox* extension
- lwIP included: **TCP-IP** communication
  - core $\leftrightarrow$ core
  - SCC $\leftrightarrow$ MCPC
  - use BSD sockets in your user space app
- **SVM** system
  - PGAS-like functions (explicit `svm_alloc()`, etc.)
  - strong and lazy release consistency models

Main Features (2)

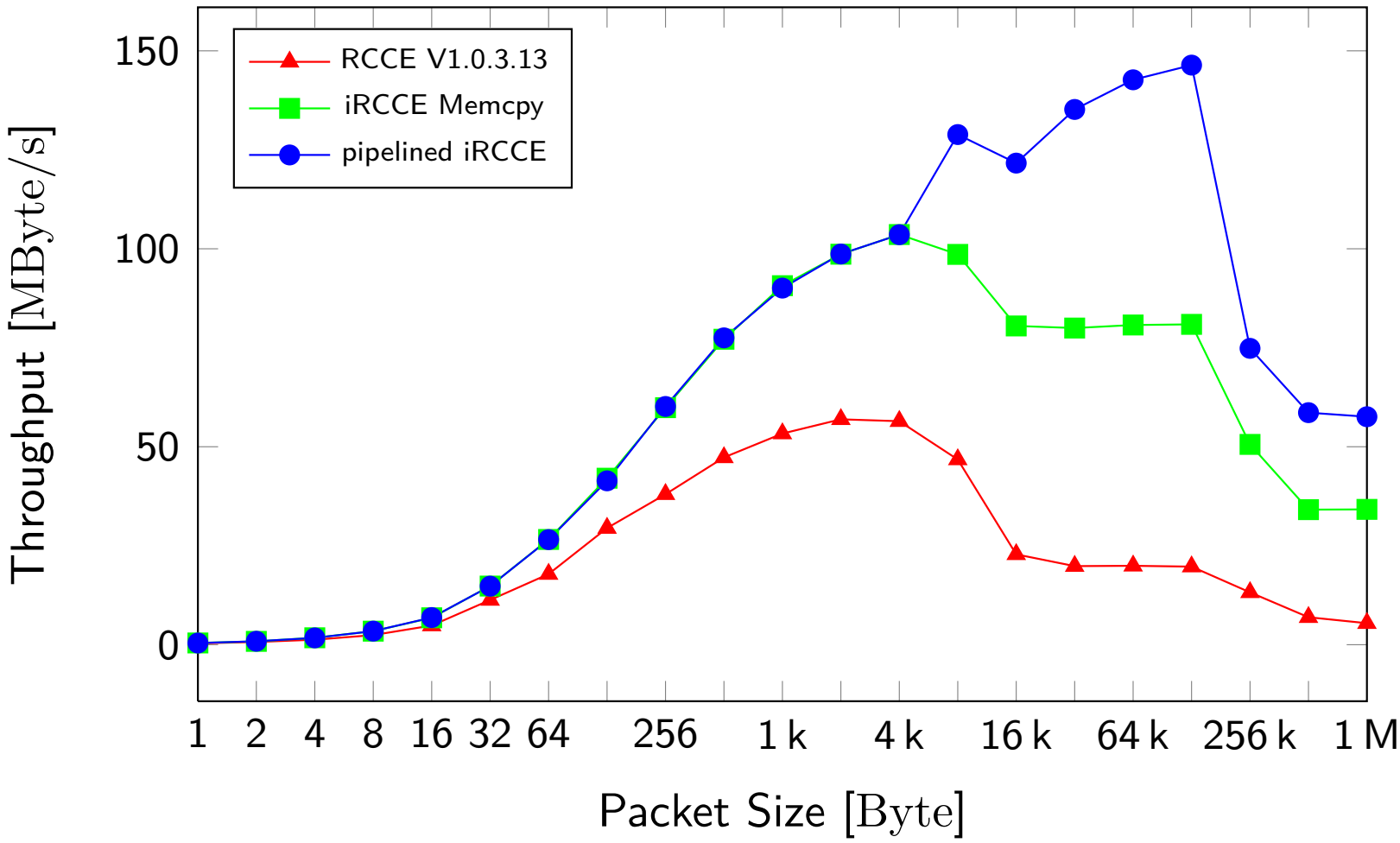
- low-latency **synchronization** layer utilizing SCC-specific *test and set-* & *atomic increment* registers
- fast scheduling
  - Round-Robin based with **priority** support
  - configurable as timeslice based or **tickless**
- device drivers
  - use your own **character devices** from user space
- basic file system
  - populated from **initial ramdisk**
  - easily customizable `/dev` interface for devices
- *newlib* **C library** for user space included
- **SMP** support (not on SCC)
- `x86_64` version in progress

Kernel Structure



kernel structure of *eduOS/MetalSVM*

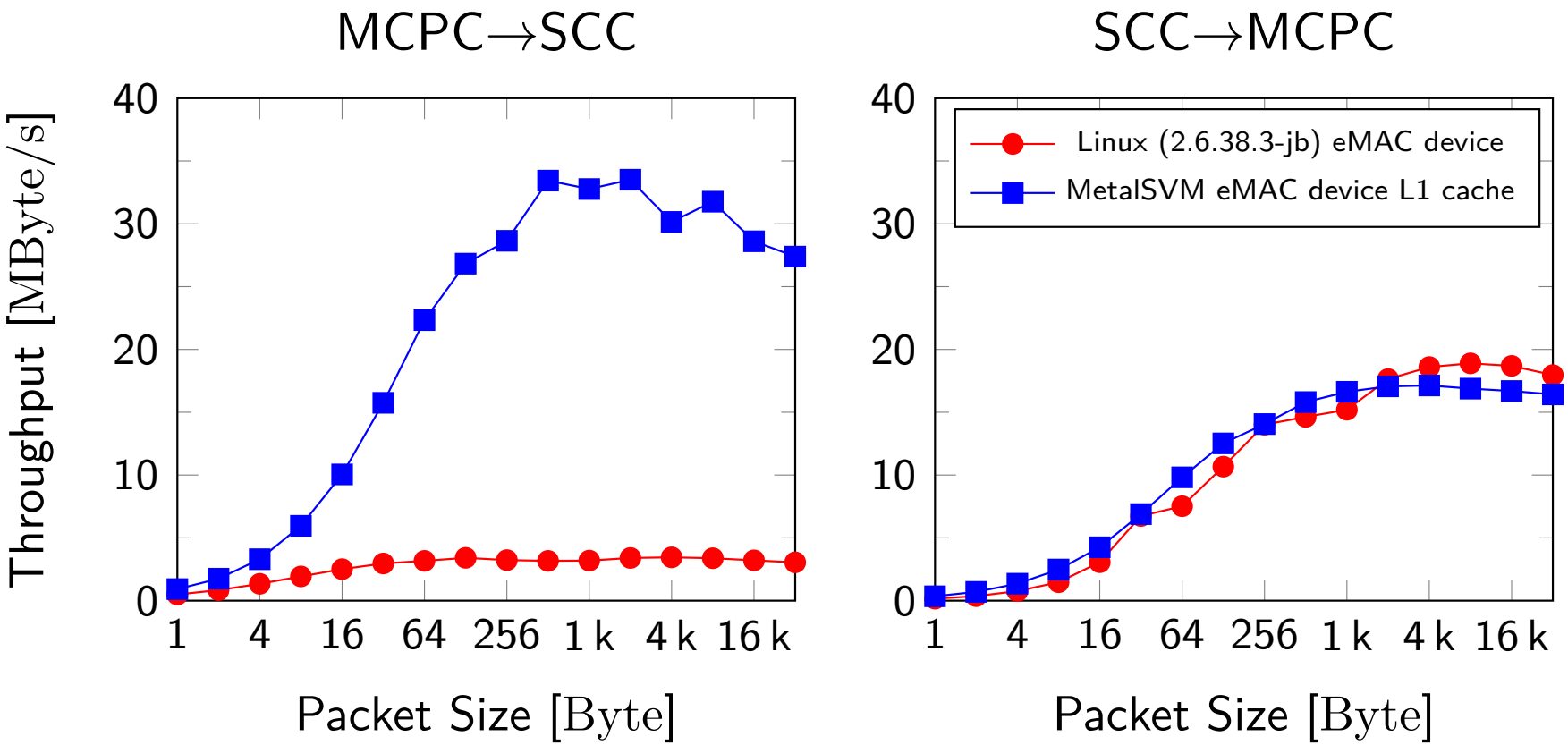
iRCCE Performance



Comparing *RCCE (V1.0.3.13)* and *iRCCE*

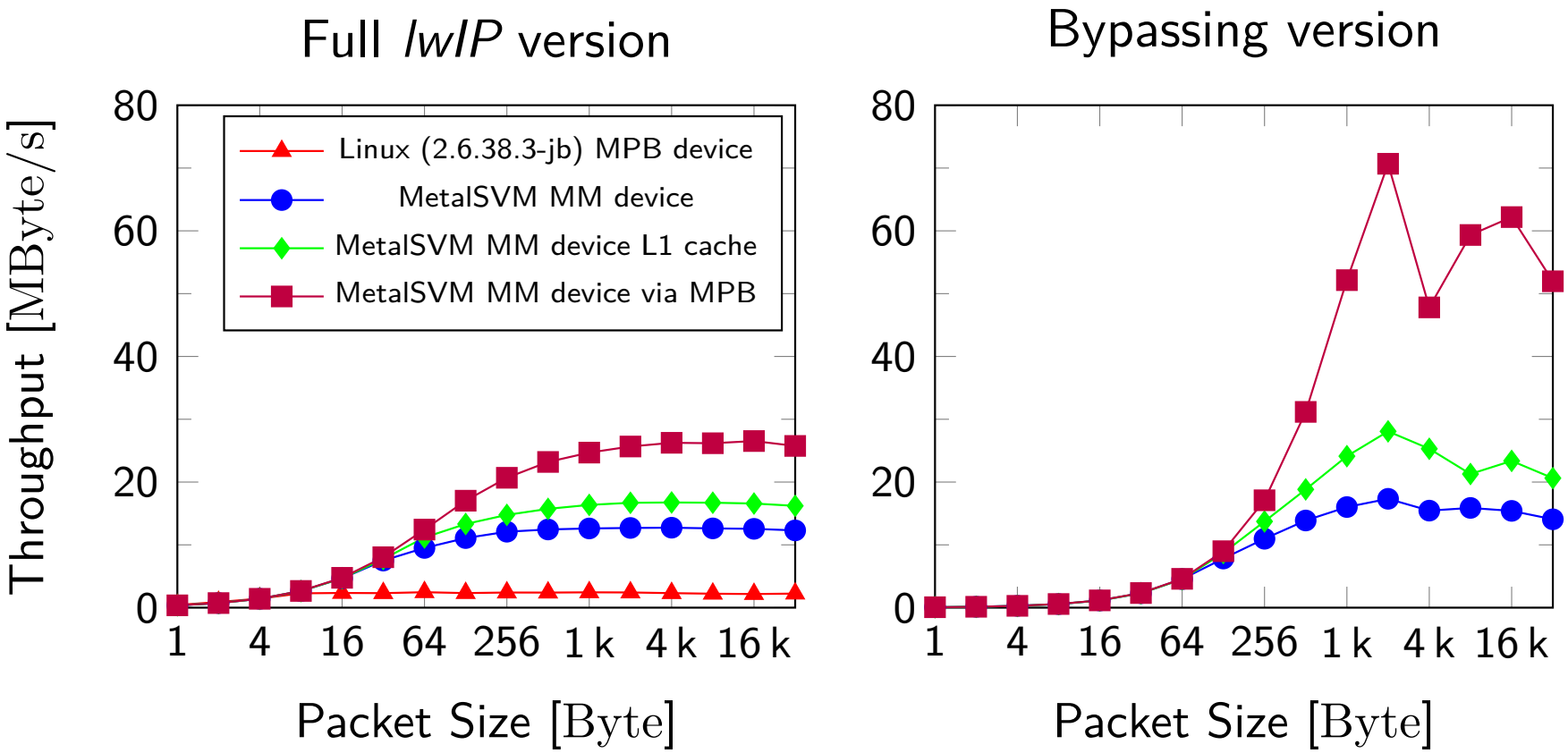
Source: iRCCE documentation

eMAC Device Driver Performance



Source: *The Path to MetalSVM: Shared Virtual Memory for the SCC*

Memory Mapped IP Driver Performance

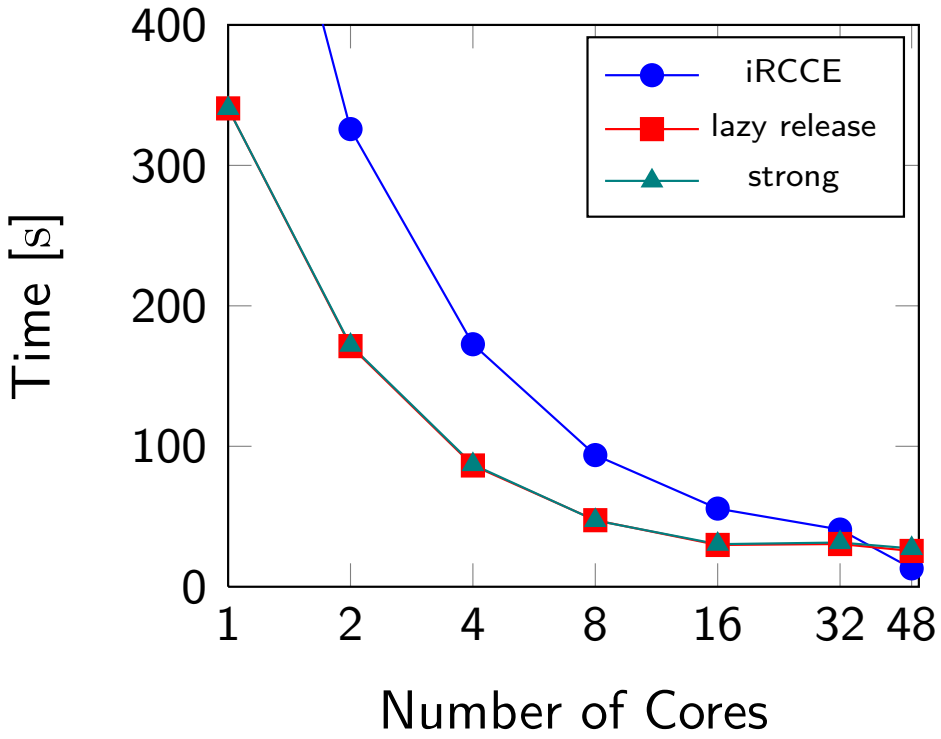


Sending packets from tile 0 to tile 1

Source: *The Path to MetalSVM: Shared Virtual Memory for the SCC*

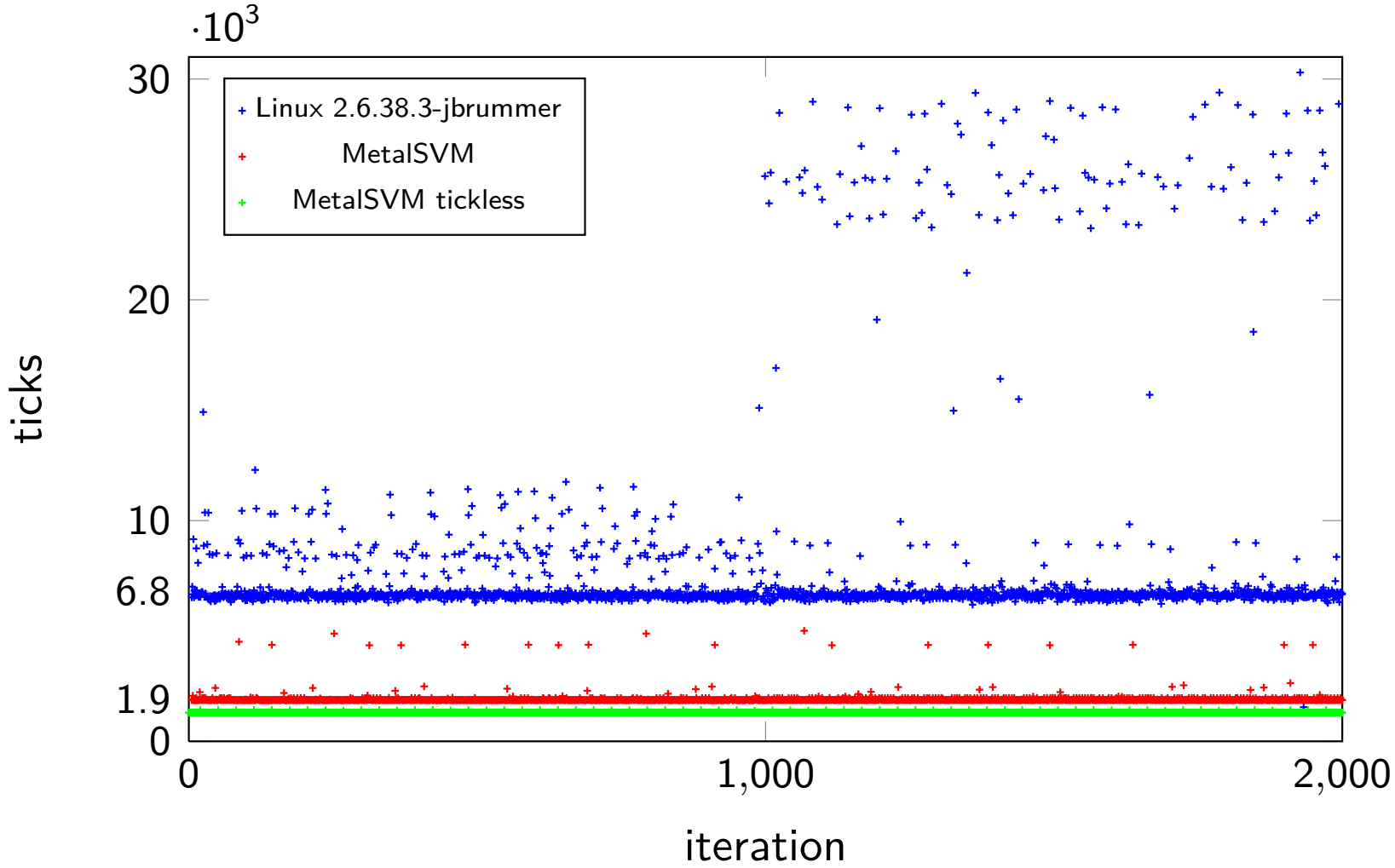
SVM Performance

- Jacobi Over Relaxation algorithm
- Problem size 1024×512
- SCC Platform running with 533 MHz core and 800 MHz memory/mesh



Source: *Revisiting Shared Virtual Memory Systems for Non-Coherent Memory-Coupled Cores*

Scheduling Overhead



Source: *Efficient Implementation of the bare-metal Hypervisor MetalSVM for the SCC*

How to Get and Use MetalSVM

Okay, I am interested!
What now?

How about getting a copy and playing with it?

Get it

Just **clone** MetalSVM from our official git repository...

```
$ git clone git://git.lfbs.rwth-aachen.de/metalsvm.git
```

...do initial **setup**...

```
$ cd metalsvm
$ cp Makefile.scc Makefile
$ (cd include/metalsvm; cp config.h.scc config.h)
```

...**build**...

```
$ make
```

...and **run** on SCC:

```
$ (cd tools; make SCC && sccBoot -g obj)
$ sccReset -r
```

Get it

Just **clone** MetalSVM from our official git repository...

```
$ git clone git://git.lfbs.rwth-aachen.de/metalsvm.git
```

...do initial **setup**...

```
$ cd metalsvm
$ cp Makefile.scc Makefile
$ (cd include/metalsvm; cp config.h.scc config.h)
```

...**build**...

```
$ make
```

...and **run** on SCC:

```
$ (cd tools; make SCC && sccBoot -g obj)
$ sccReset -r
```

Get it

Just **clone** MetalSVM from our official git repository...

```
$ git clone git://git.lfbs.rwth-aachen.de/metalsvm.git
```

...do initial **setup**...

```
$ cd metalsvm
$ cp Makefile.scc Makefile
$ (cd include/metalsvm; cp config.h.scc config.h)
```

...**build**...

```
$ make
```

...and **run** on SCC:

```
$ (cd tools; make SCC && sccBoot -g obj)
$ sccReset -r
```

Get it

Just **clone** MetalSVM from our official git repository...

```
$ git clone git://git.lfbs.rwth-aachen.de/metalsvm.git
```

...do initial **setup**...

```
$ cd metalsvm
$ cp Makefile.scc Makefile
$ (cd include/metalsvm; cp config.h.scc config.h)
```

...**build**...

```
$ make
```

...and **run** on SCC:

```
$ (cd tools; make SCC && sccBoot -g obj)
$ sccReset -r
```

Get it (2)

Connect and send commands:

```
$ telnet rck00 4711
```

Build documentation:

```
$ doxygen
```

→ see `documentation/html/index.html`

Configure it

`include/metalsvm/config.h`

configure if you want to include support for
PCI, lwIP, VGA, software-UART, Keyboard, Multiboot/SCC,
change timeslice size, enable tickless scheduling, etc.

`Makefile`

Depending on wether you run MetalSVM in qemu/on SCC
$\rightarrow$ set compiler, etc.

Customize it

The most important files and directories

`apps/`	kernel space apps
`newlib/examples/`	user space apps
`apps/tests.c`	the init “daemon”
`drivers/char/`	character device drivers
`tools/`	mostly SCC-specific build stuff

Conclusion

With MetalSVM as a framework for **your** research on the SCC, you get:

- **fast**, bare-metal like execution of your code
- basic OS **comfort** in kernel- and user space
- fast communication:
  - **TCP-IP** & **iRCCE**
  - blocking & non-blocking, time- & event driven
  - core $\leftrightarrow$ core, SCC $\leftrightarrow$ MCPC
- **SVM** support
- a kernel which is easily **customizable** to your needs

Finally

check out

http://metalsvm.org

contact@metalsvm.org

Questions?