IPDPS 2010 @ Atlanta

GenerOS: An Asymmetric Operating System Kernel for Multi-core Systems

Authors:
Qingbo Yuan, Jianbo Zhao, Mingyu Chen, Ninghui Sun
Institute of Computing Technology
Chinese Academy of Sciences
yuanbor@ncic.ac.cn

Speaker:
SuZhen Wu
Huazhong University of Science & Technology

Outline

1 Motivation

2 Architecture of GenerOS

3 Implementation of GenerOS

4 Evaluation of GenerOS vs Linux

5 Conclusion

Motivation

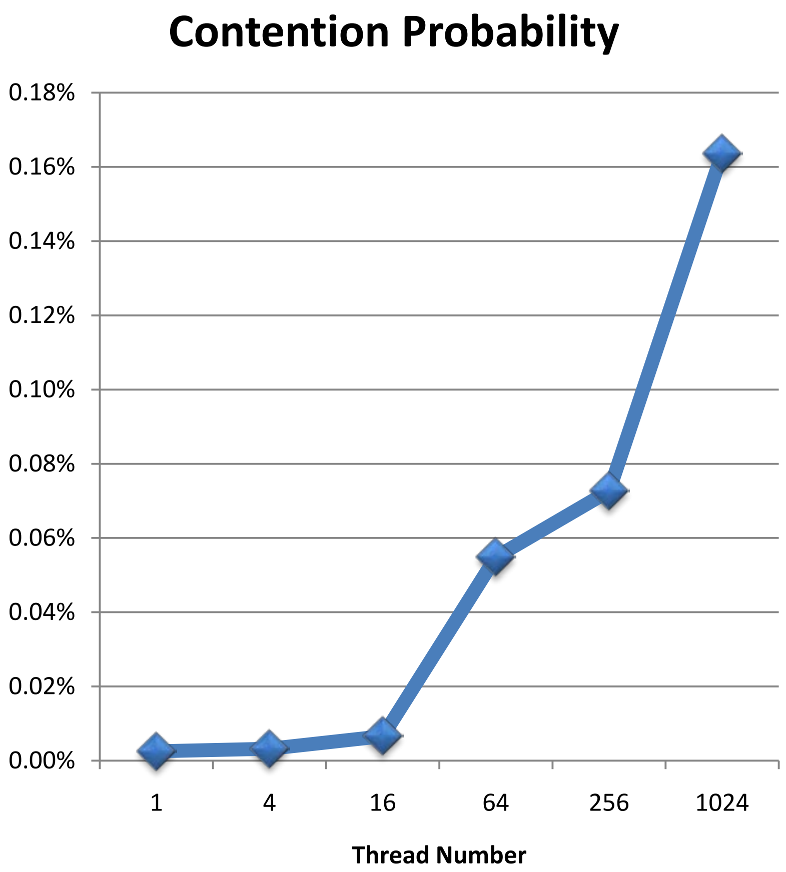
- Symmetric multithread operating system such as Linux suffers from lock contention and cache pollution
- Lock contention
  - As more cores are packaged into a single chip, there are two many cores in a system
  - Each core has the ability to trap into kernel
  - Too many procedures in kernel -> serious lock contention
- Cache pollution
  - Applications and kernel run on the same core
  - Applications may kick kernel’s cache line out of cache
  - And vice versa

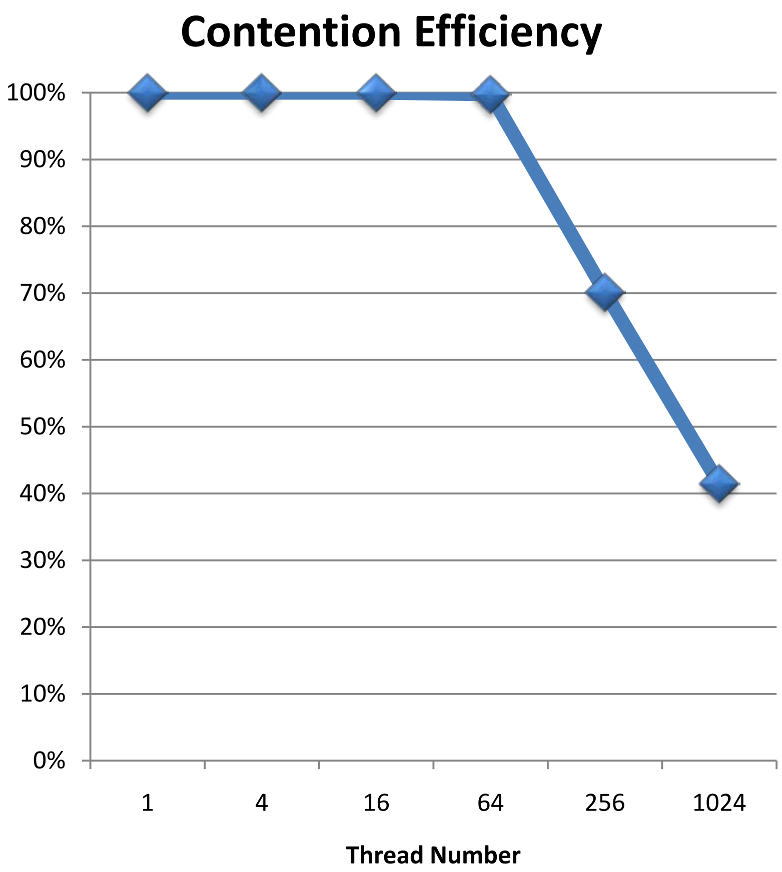
Motivation
---- Lock Contention @ Linux

- **Contention Probability** = contentions / acquisitions
  - acquisitions: times acquiring lock
  - contentions: times encountering contention

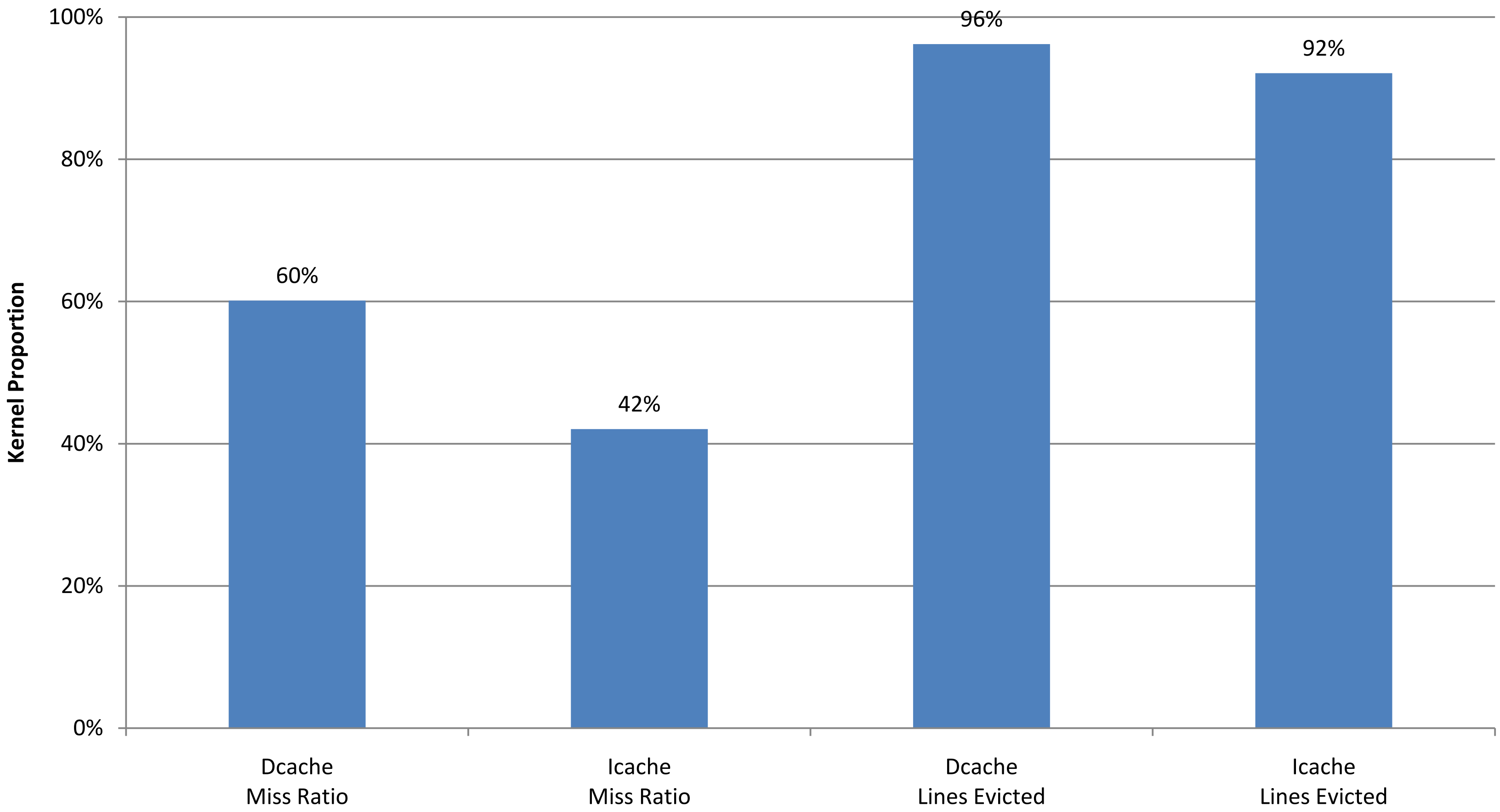
- **Contention Efficiency** = hold time / (hold time + wait time)
  - hold time: time in critical region
  - wait time: time waiting for entering critical region

Motivation
---- Lock Contention @ Linux

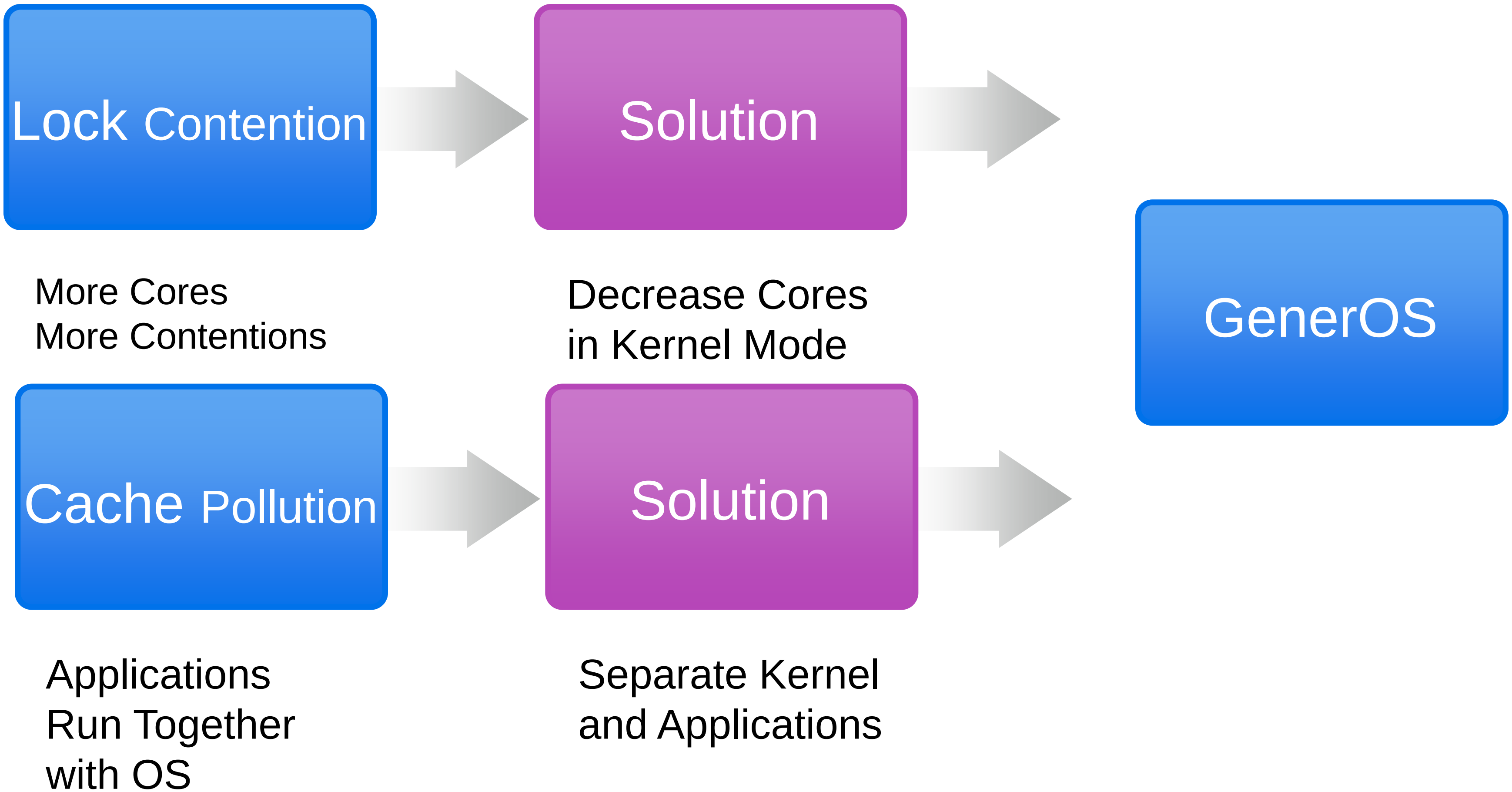




Motivation
---- Cache Pollution @ Linux



Motivation


Lock Contention
Solution
More Cores
More Contentions
Decrease Cores
in Kernel Mode
GenerOS
Cache Pollution
Solution
Applications
Run Together
with OS
Separate Kernel
and Applications

Outline

1 Motivation

2 Architecture of GenerOS

3 Implementation of GenerOS

4 Evaluation of GenerOS vs Linux

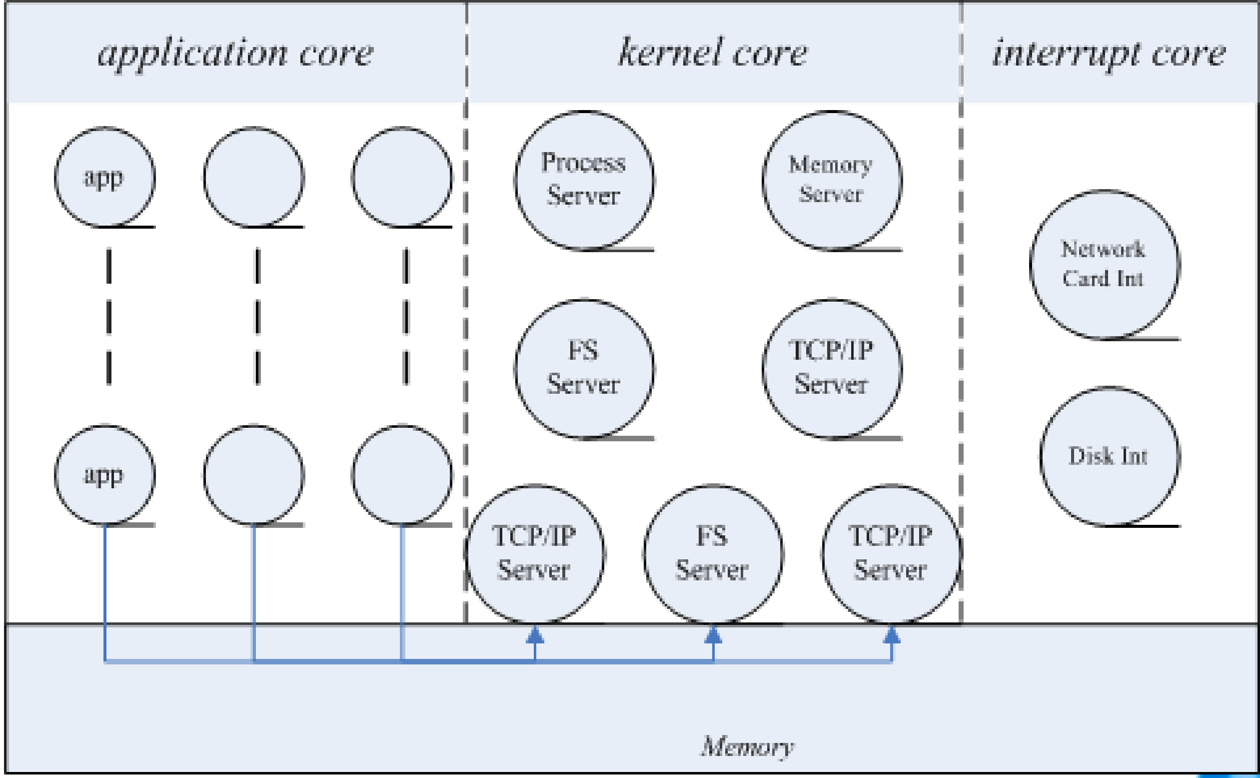
5 Conclusion

Architecture

- In a symmetric multiprocessing system, Linux treats all cores as an equal which causes a lot of problems

- By contrast, GenerOS partitions processing cores into application core, kernel core and interrupt core
  - All of applications run on application core
  - Their system calls are executed by kernel core
  - Interrupts are all bound to interrupt core

Architecture



- Most of cores are used by applications
- A limited number of cores are used by kernel service
  - File System
  - Process
- Few number of cores are used to handle interrupt

Outline

1 Motivation

2 Architecture of GenerOS

3 Implementation of GenerOS

4 Evaluation of GenerOS vs Linux

5 Conclusion

Implementation

- GenerOS is developed based on Linux-2.6.25 @ x86_64 architecture
- In system call level, several kernel servers are developed
  - File system server (98 system calls)
    - sys_open / sys_close / sys_read / sys_write
  - Network server (15 system calls)
    - sys_socket / sys_connect
  - Signal server (12 system calls)
    - sys_rt_sigaction
  - IPC server (12 system calls)
    - sys_msgget
  - Process server (10 system calls)
    - sys_fork
  - Others (141 system calls)
    - sys_brk

Implementation
---- GenerOS Processing Flow Chart

Application @ Application core

```
int main(void)
{
    pid_t getpid();

    return 0;
}
```

Runtime @ Application core

```
pid_t generos_sys_getpid(void)
{
    req = generos_get_request();
    generos_init_request(req);
    generos_send_to_kernel(req);
    sleep();
    return pid;
}
```

Process Server @ Kernel core

```
while(generos_request_queue_is_not_empty(&process_queue)){
    req = generos_pick_request(&process_queue);
    switch(req->type){
        case GETPID:
            req->retvalue = sys_getpid();
            break;
        ……
    }
    wake_up_process(req->task);
}
```

Implementation
---- Runtime at Application Core

- It replaces the system call table of Linux

```
const sys_call_ptr_t syscall_table [__NR_syscall_max+1]  = {
        [__NR_read] = &generos_sys_read,
        [__NR_write] = &generos_sys_write,
        ……
        [__NR_timerfd_gettime] = &generos_sys_timerfd_gettime;
};
```

- The left side keeps the same meaning with Linux which makes GenerOS compatible with Linux
- The right side uses self defined function which will find a kernel core to handle its system call

Implementation
---- Kernel Core

- Two queues
  - Request queue
    - Receive system call requests from application core
  - Wait queue
    - Store the being handled system calls which are waiting for something
- One schedule method
  - Slim Schedule
    - Schedule system calls in this kernel core with almost zero overhead

Implementation
---- Binding Interrupt Handler

- Interrupt core is used to deal with most of interrupts from network interface, disk, or local timer
- In such way, both of application core and kernel core will have a clean execution environment
- GenerOS uses the method in Linux to bind interrupt handler to some processing core

Outline

1 Motivation

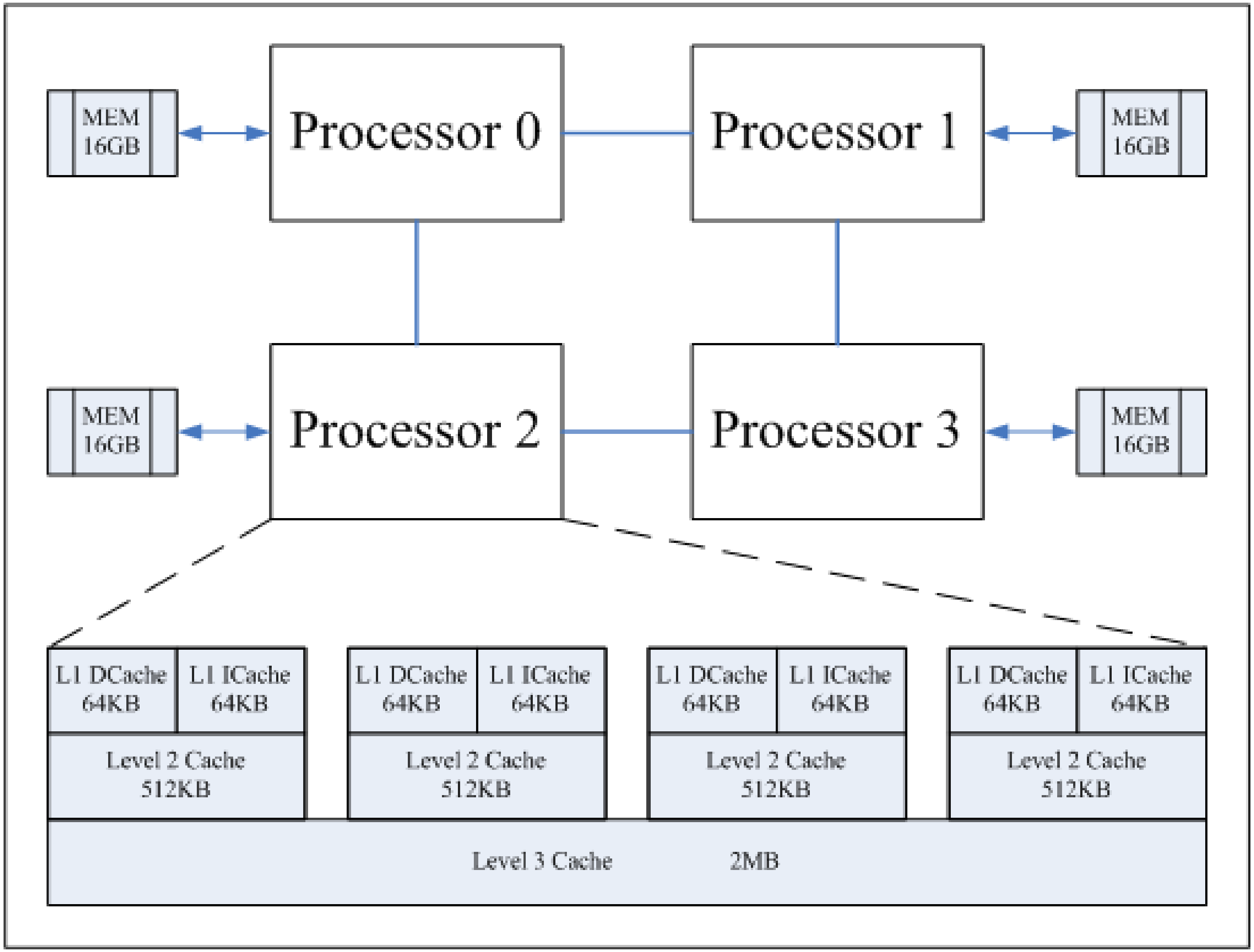
2 Architecture of GenerOS

3 Implementation of GenerOS

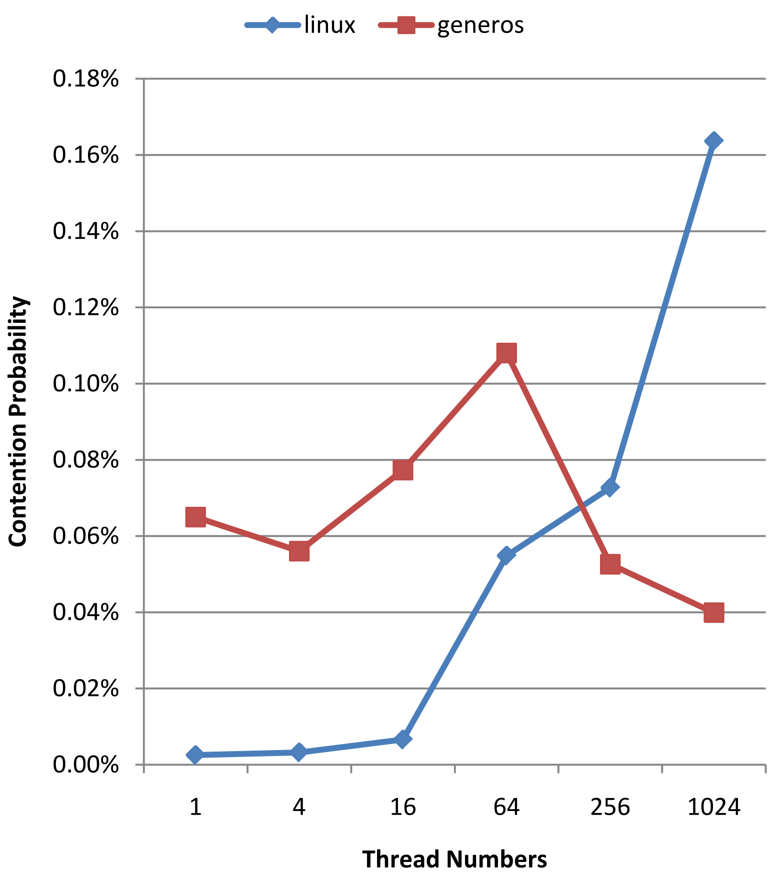
4 Evaluation of GenerOS vs Linux

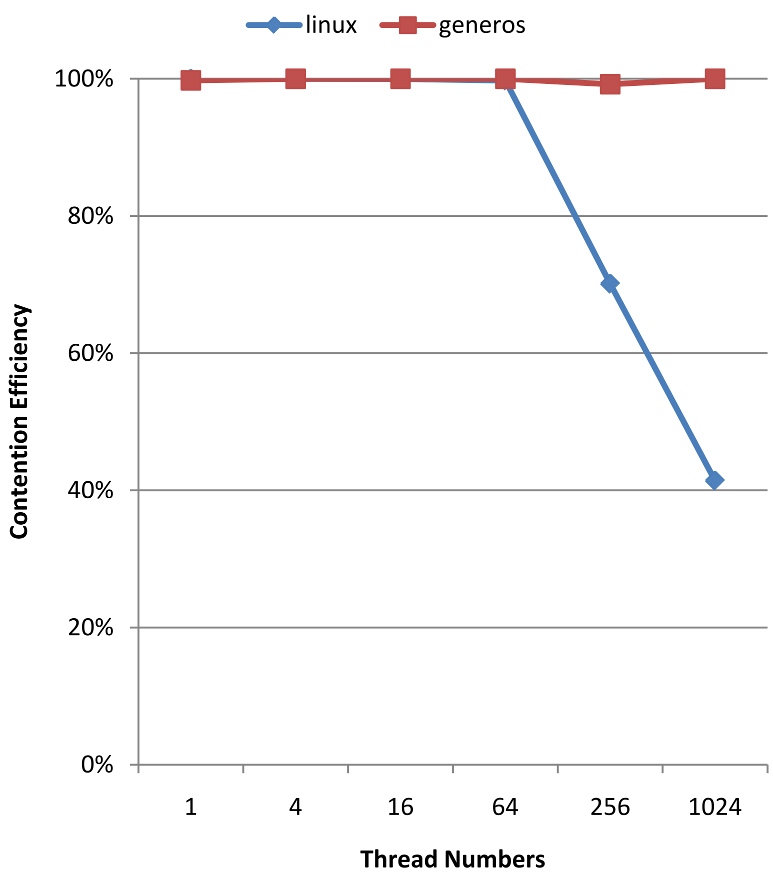
5 Conclusion

Evaluation
---- Platform

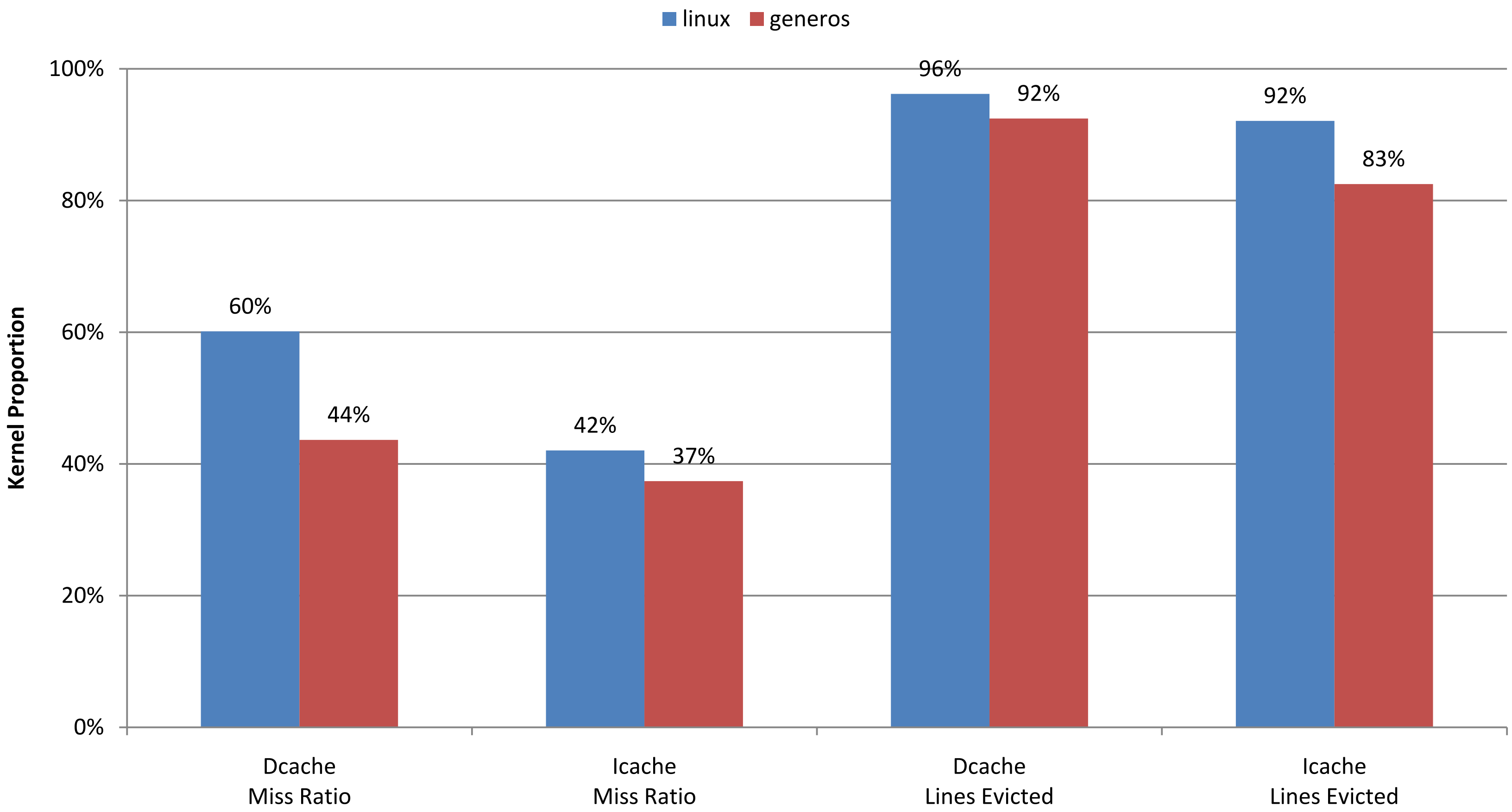


Evaluation
---- Lock contention

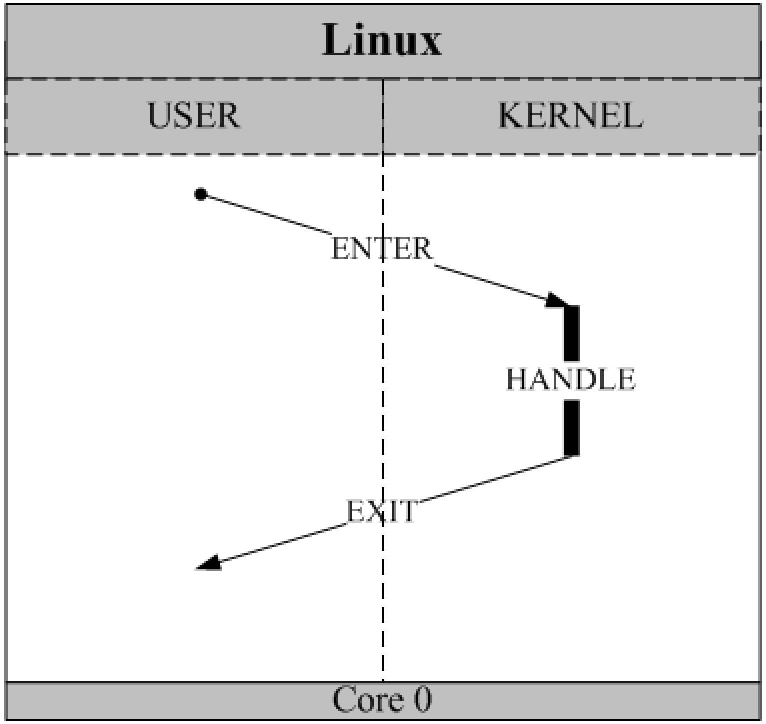


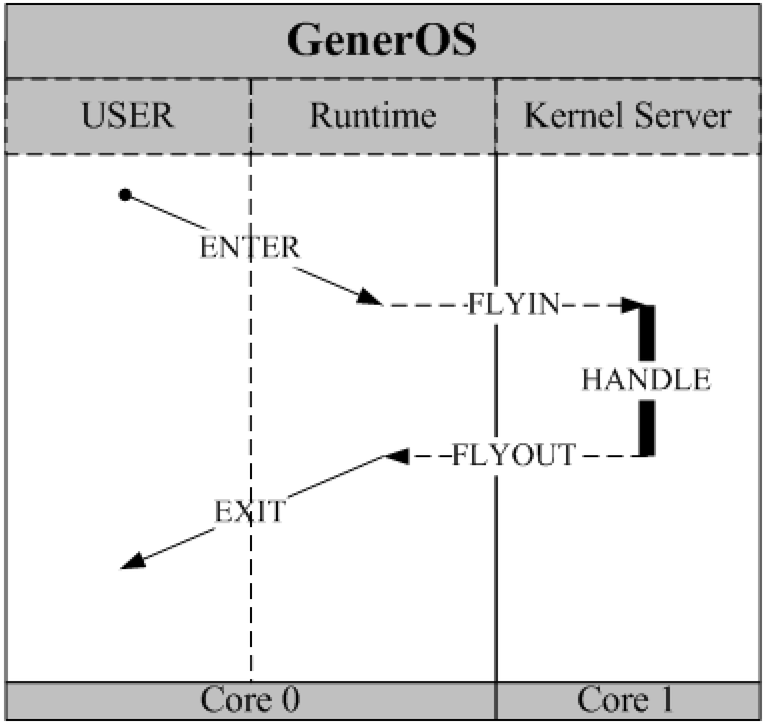


Evaluation
---- Cache Pollution

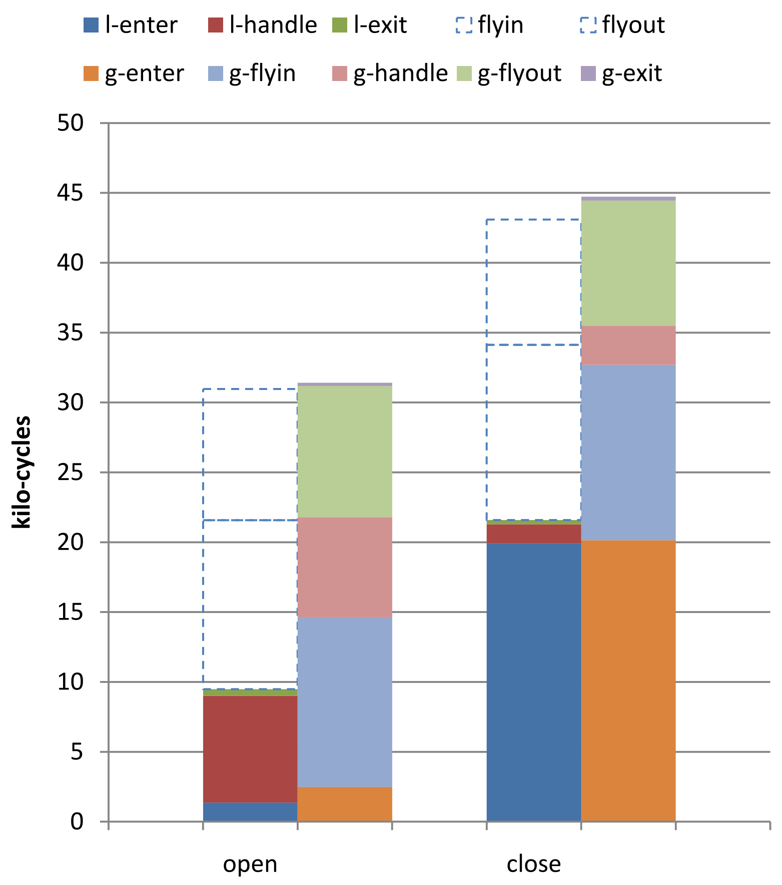


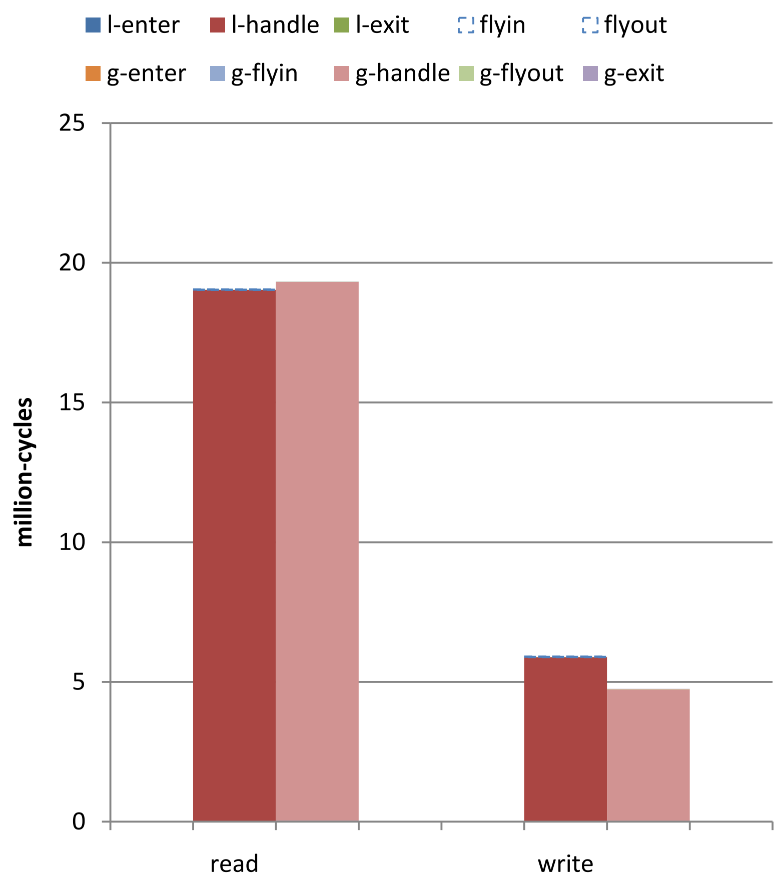
Evaluation
---- Single System Call



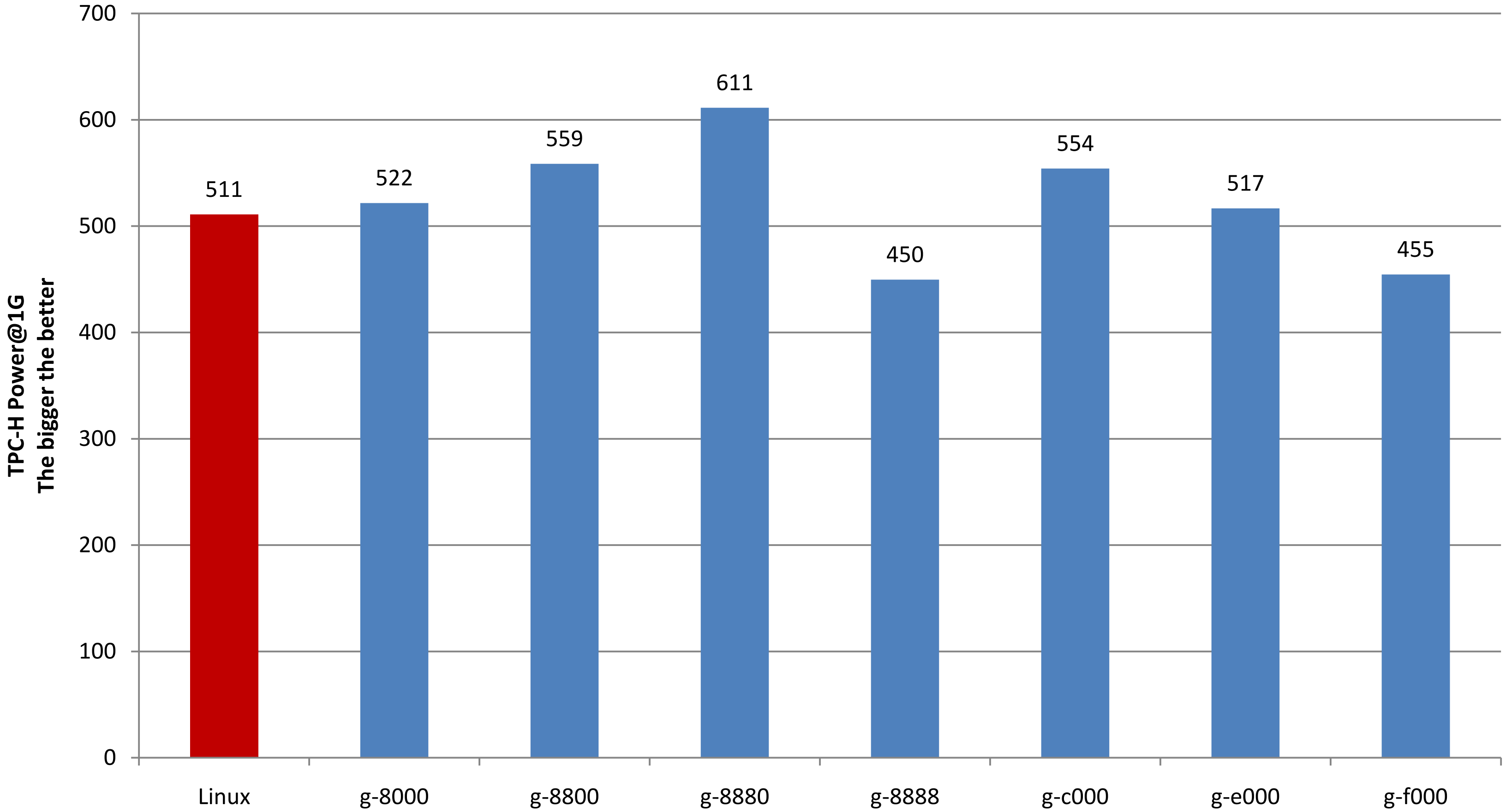


Evaluation
---- Single System Call





Evaluation
---- TPC-H 1GB Power

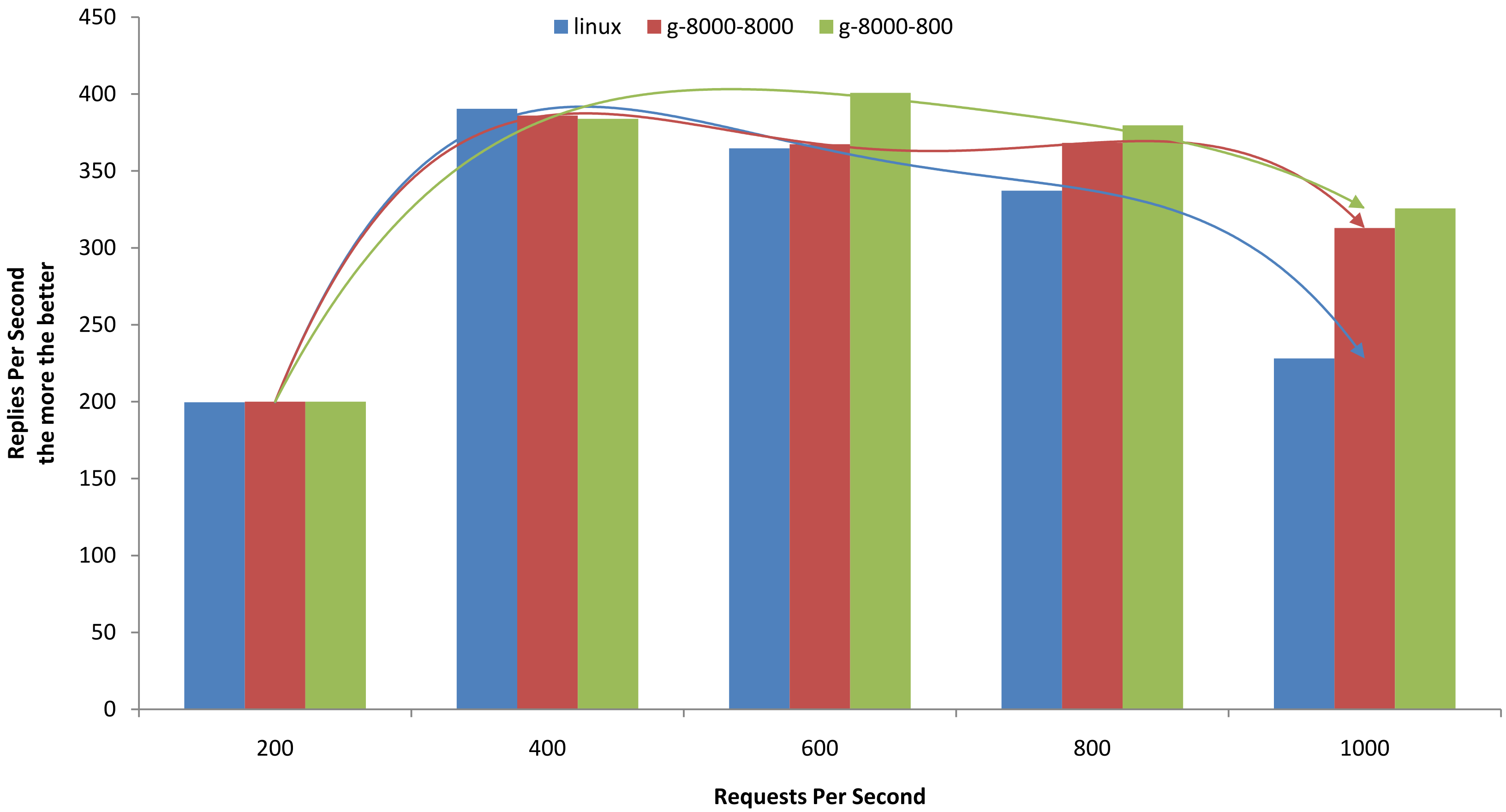


Evaluation
---- TPC-H 1GB Power

Table 1: GenerOS Configuration

	application core	kernel core	interrupt core
g-8000	0 ~ 14	15	15
g-8800	0 ~10, 12~14	11, 15	11, 15
g-8880	0~6, 8~10,12~14	7, 11, 15	7, 11, 15
g-8888	0~2, 4~6, 8~10, 12~14	3, 7, 11, 15	3, 7, 11, 15
g-c000	0~13	14, 15	14, 15
g-e000	0~12	13, 14, 15	13, 14, 15
g-f000	0~11	12, 13, 14, 15	12, 13, 14, 15

Evaluation
---- Httperf



Outline

1 Motivation

2 Architecture of GenerOS

3 Implementation of GenerOS

4 Evaluation of GenerOS vs Linux

5 Conclusion

Conclusion

- GenerOS is an asymmetric kernel which is designed to deal with the problems faced in traditional symmetric kernel
- Being compatible with Linux, GenerOS does not need to modify, recompile, or relink applications, or libraries
- Experiments with two typical workloads on 16-core AMD machine show that GenerOS behaves better than original Linux kernel when there are more processing cores
  - 19.6% for TPC-H using oracle database management system
  - 42.8% for httperf using apache web server

Thank you very much!

Any question ?

Please contact the author

yuanbor@ncic.ac.cn