Static Linking VS. Dynamic Linking

Haitao Wang, Xiaomin Liu

December 13, 2006

Static Linking

Carried out only once to produce an executable file.

If static libraries are called, the linker will copy all the modules referenced by the program to the executable.

Dynamic Linking

Allows a process to add, remove, replace or relocate object modules during its execution.

If shared libraries are called,

1. Only copy a little reference information when the executable file is created.
2. Completing the linking during loading time or running time.

A Big Difference

If several processes call the same object module of a shared library simultaneously,

- Only one copy in memory (dynamic)
- Several copies each for a process in memory (static)

Which one is better, static or dynamic?

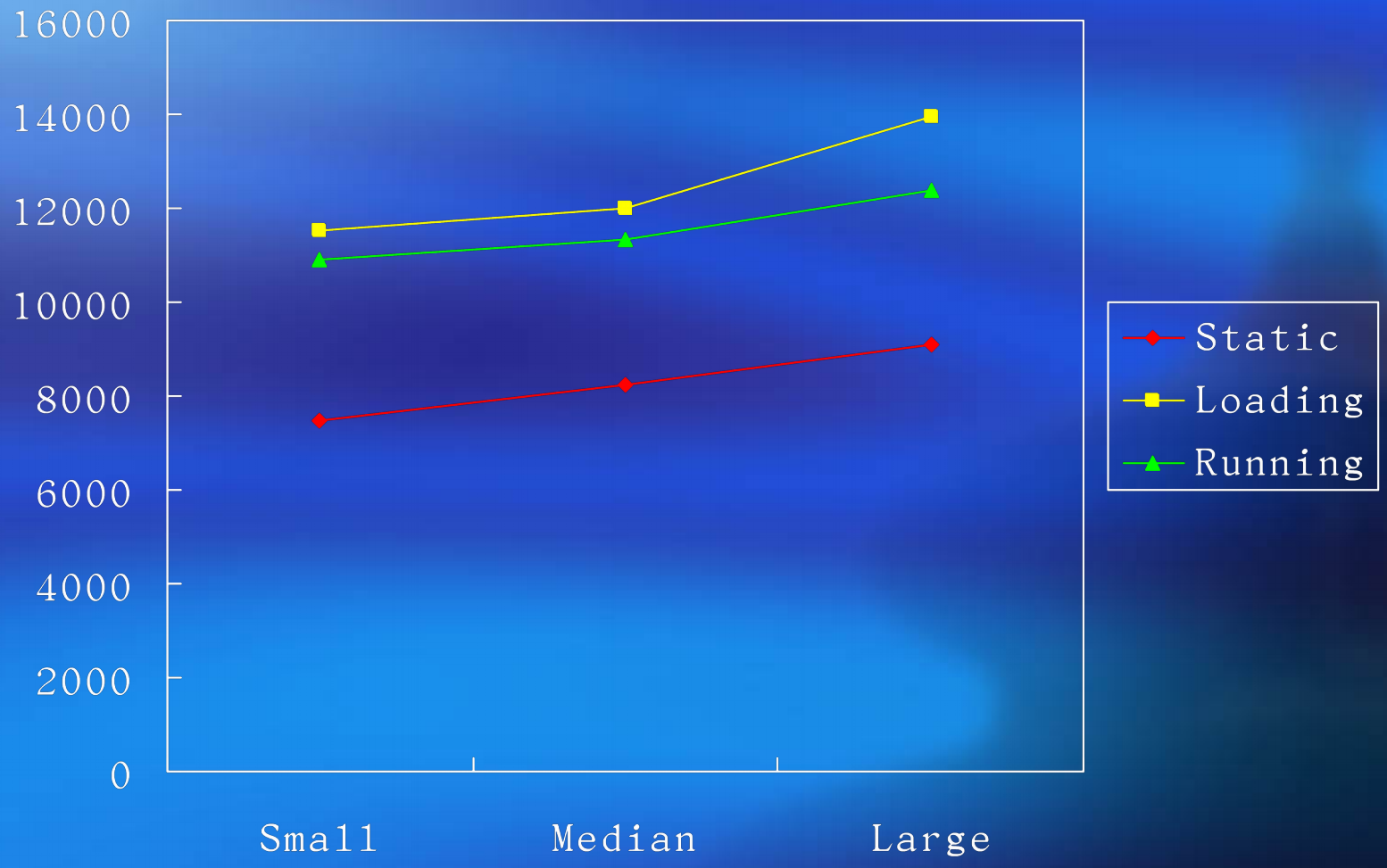
Construct some programs of different size linked in both versions to compare:

- Executable size
- Loading time
- Running time
- System call
- Memory usage

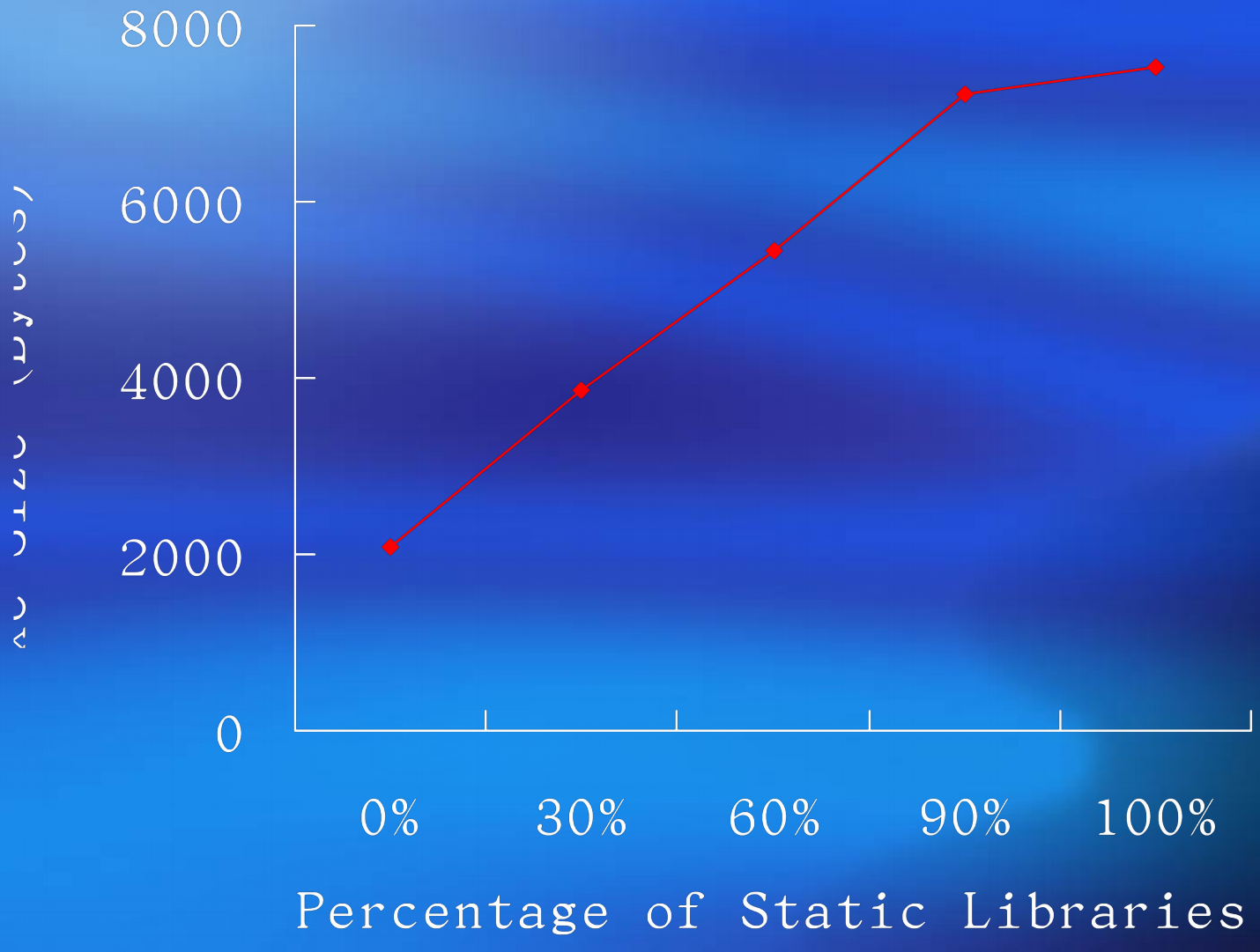
Executable size (Byte)

	Static	Dynamic (Loading)	Dynamic (Running)
Small	422596	5152	6300
Median	431413	5152	6297
Large	443384	5134	6312

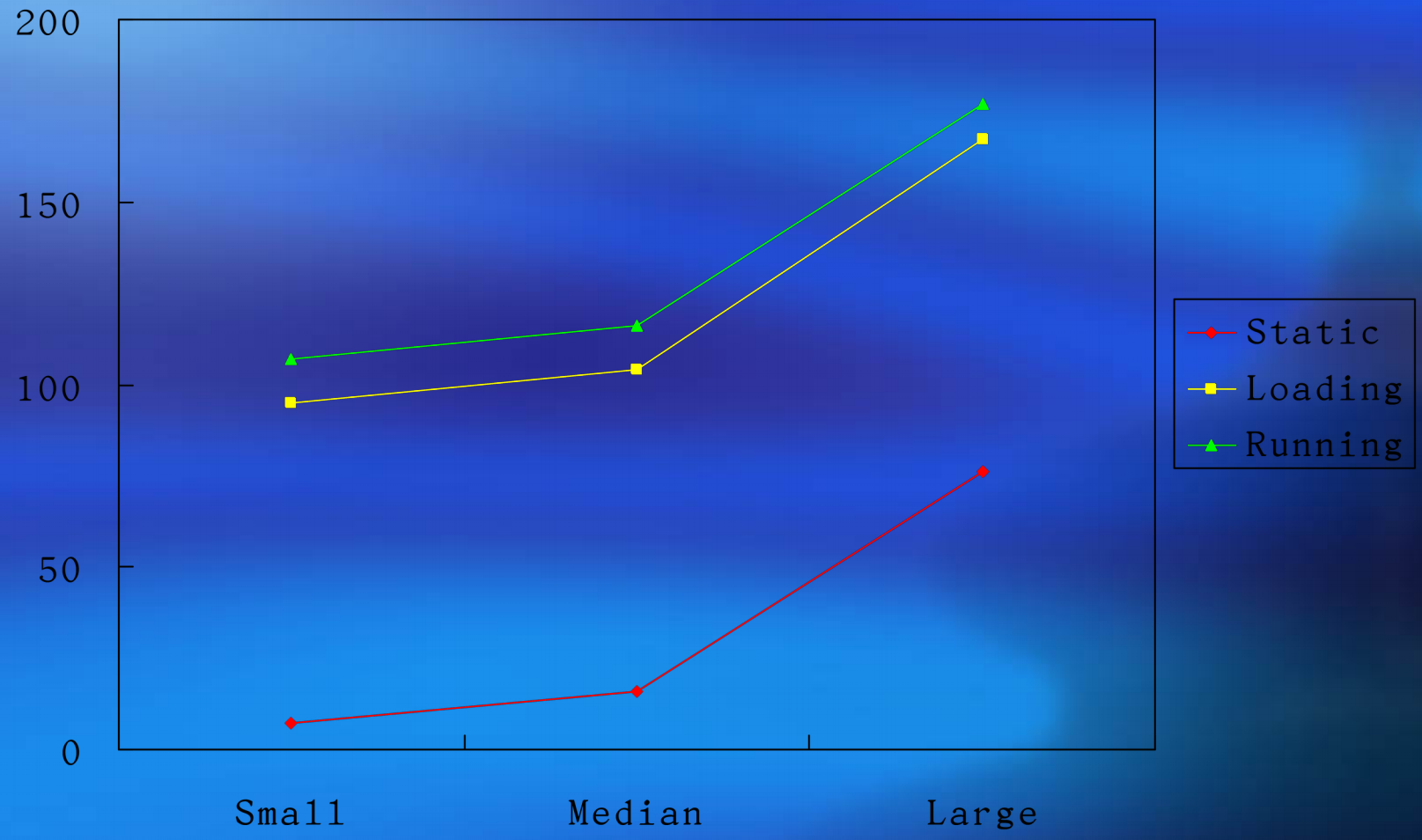
Loading time (us)



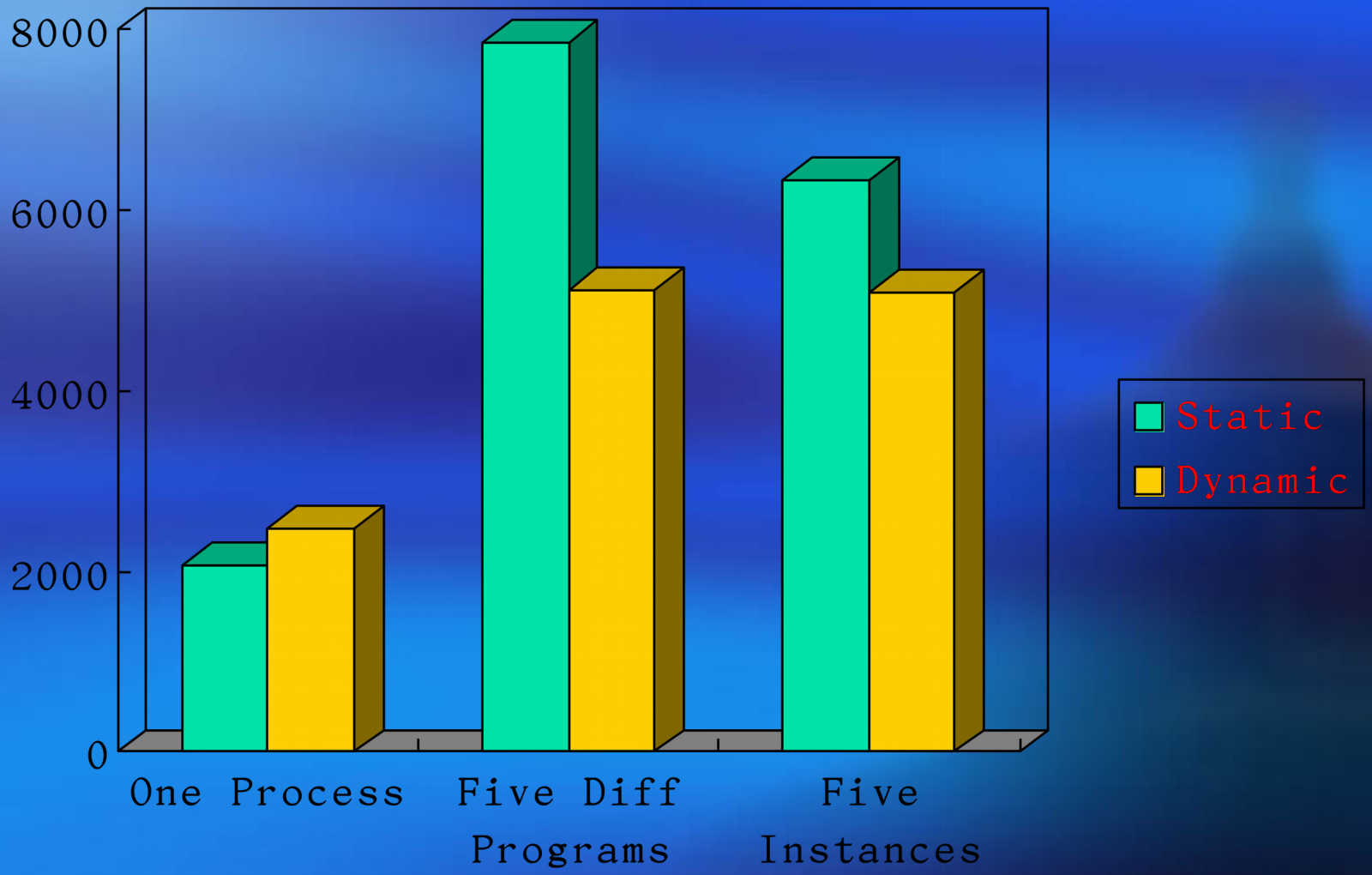
Running time (s)



Number of System Calls



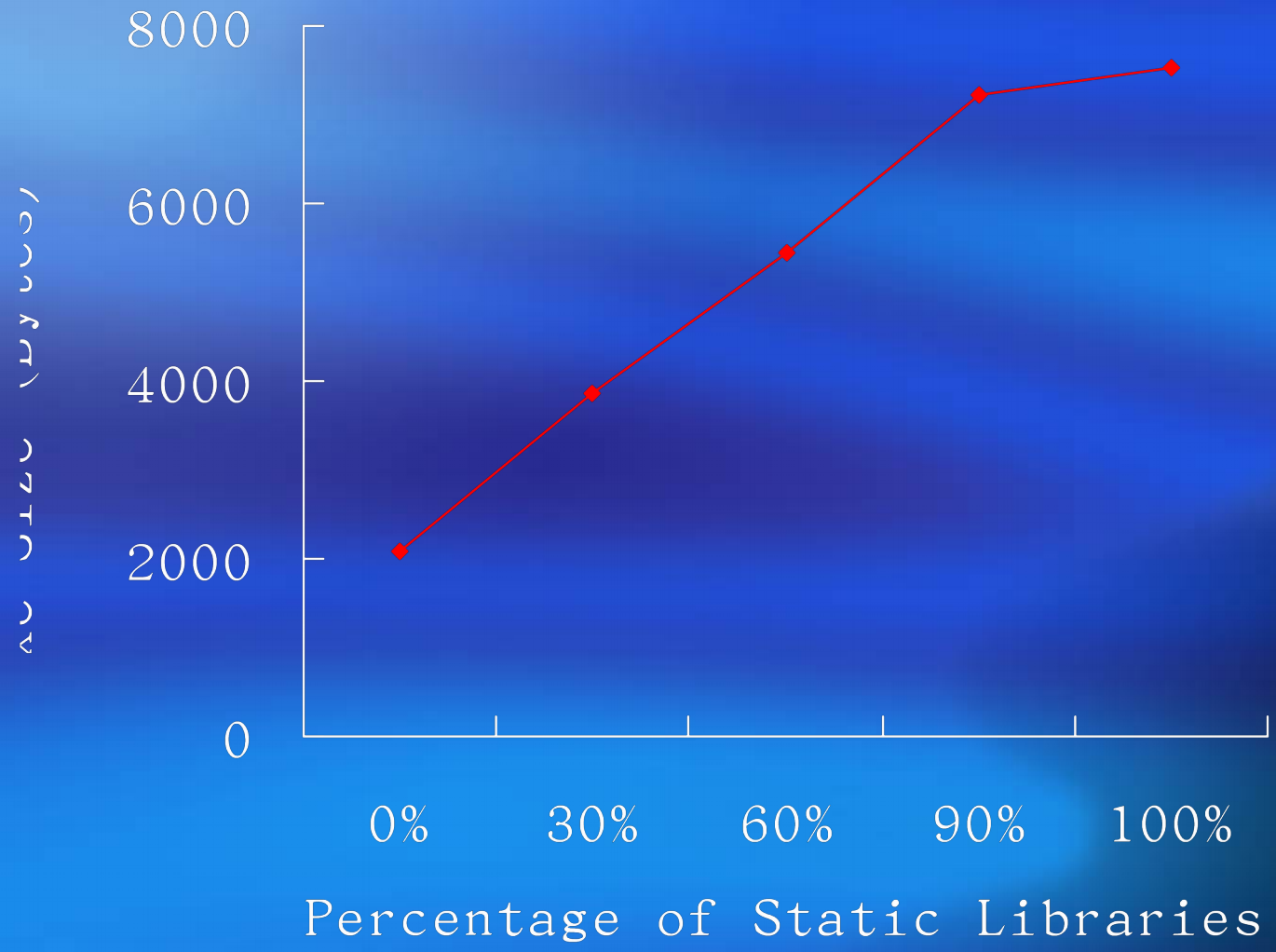
Memory Usage (Byte)



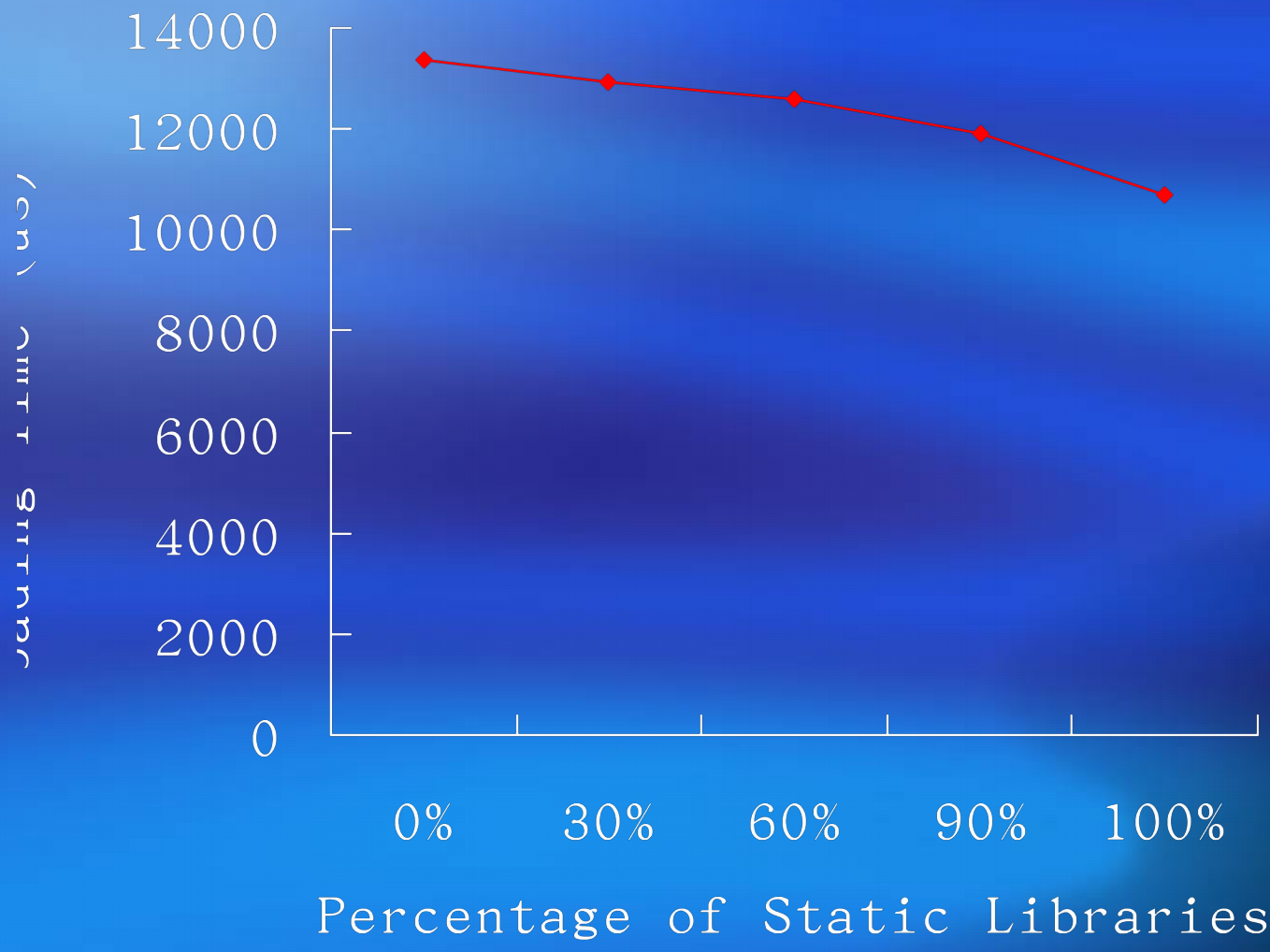
Change the Percentage of Static Lib

Produce several different versions of executables by changing the percentage of the static libraries among all libraries written by ourselves when programs are linked (those standard libs, such as libc, are always dynamically linked).

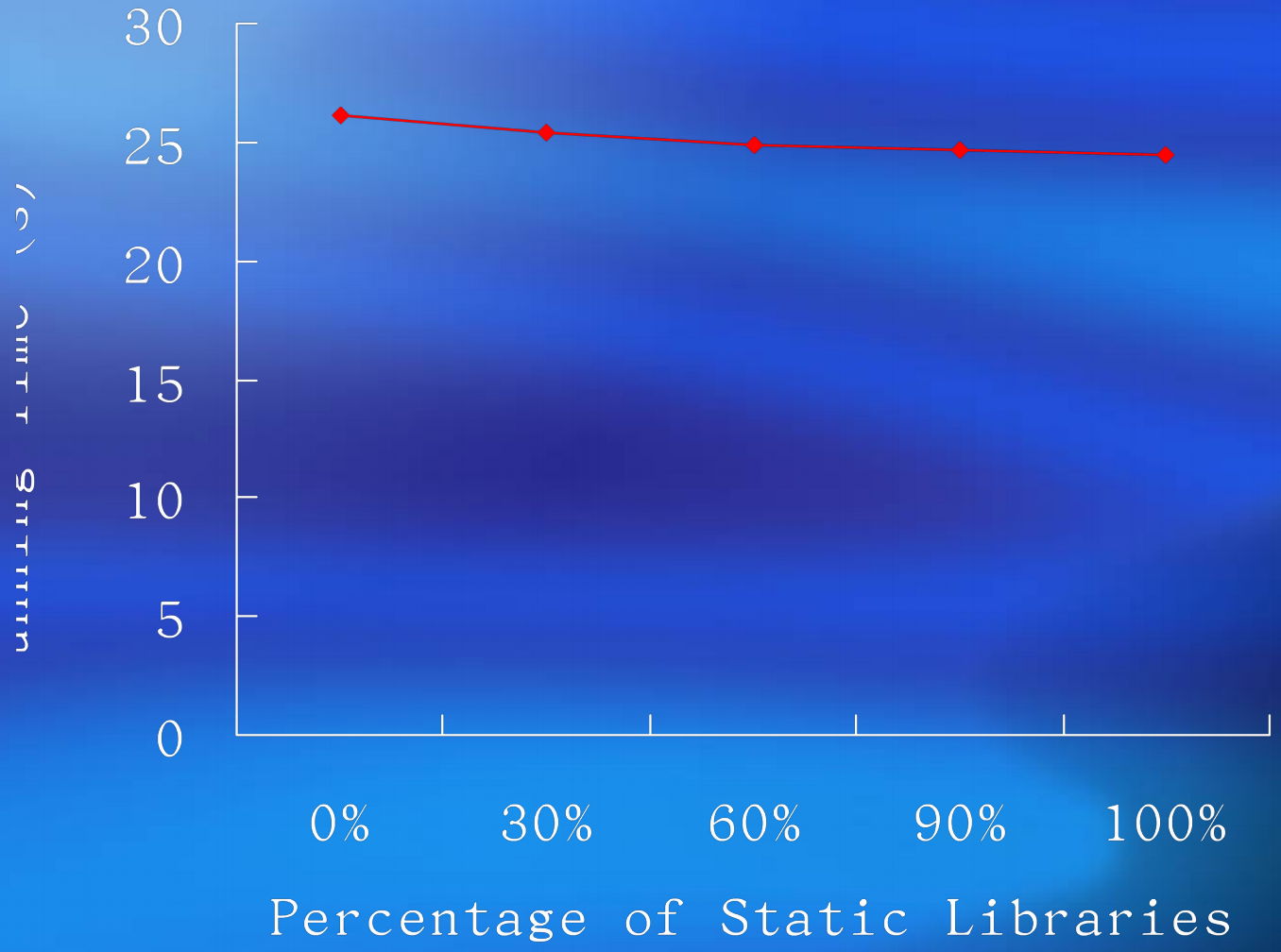
Executable Sizes



Loading Time



Running Time



Static linking VS. Dynamic linking

- Dynamic produces smaller executable files.
- Dynamic consumes less memory.

- Dynamic runs more slowly.
- Dynamic has more system calls.
- Dynamic need more loading time

Standard Programs

- Some programs in Linux system are rebuilt. Both static and dynamic versions are created and some benchmarks are compared.

- PMEM, MPG123, FGET, VIM

Executable Size (Bytes)

	PMEM	MPG	FGET	VIM
Static	46043	979931	582796	9935633
Dynamic	17892	494443	93721	3875807

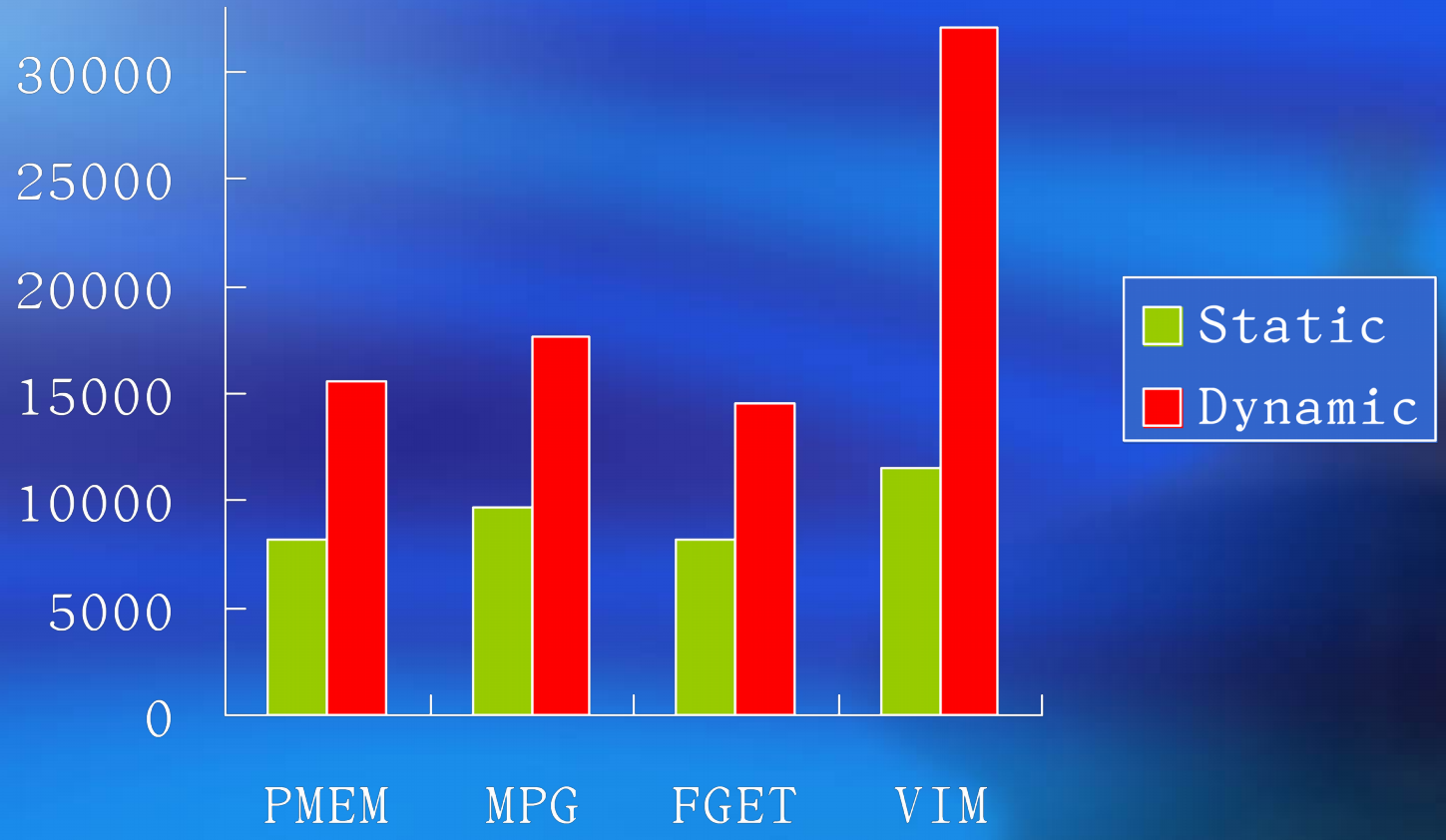
Number of System Calls

	PMEM	MPG	FGET	VIM
Static	5	34	11	1626
Dynamic	102	178	119	2218

Vim System Call

Sys call	Open	Close	Read	Stat64	Old_mmap
Static	88	58	473	11	40
Dynamic	296	102	569	55	110

Loading Time



Sharing Across File System

- Examine the contents of executables and libraries in a standard Linux system.
- Programs in /bin and /usr/bin are examined.

- There are 1831 dynamically linked programs and only one static linked (/bin/ash.static), which also has its dynamic version (/bin/ash) in the system.

- Dynamic linking is dominant throughout the system.

Most Frequently Referenced Libs

Name	Directory	Prog Num
libc.so.6	/lib/tls/libc.so.6	1831
ld-linux.so.2	/lib/ld-linux.so.2	1831
libm.so.6	/lib/tls/libm.so.6	1084
libdl.so.2	/lib/libdl.so.2	684
libz.so.1	/usr/lib/libz.so.1	596
libpthread.so.0	/lib/tls/libpthread.so.0	569
libresolv.so.2	/lib/libresolv.so.2	438
libX11.so.6	/usr/X11R6/lib/libX11.so.6	424
libXext.so.6	/usrX11R6/lib/libXext.so.6	412

Programs with Most Libs

Program Name	Number of Libraries
Ximian-connector-setup-2.0	87
Evolution-2.0	75
Nautilus	70
Nautilus-file-management-properties	70
Srcore	70
Gnomemeeting	68
Create-branching-keyboard	67
Devhelp	67
Gok	67

Future Work

- Build some much larger standard programs in both versions and compare them.
- Construct an entire system distribution both statically and dynamically to evaluate the performance.