The Performance Evaluation of Omni OpenMP Compiler

Kazuhiro Kusano
Shigehisa Satoh Mitsuhisa Sato

Real World Computing Partnership
Tsukuba research center

Background

OpenMP is proposed as a parallel programming interface for shared memory machines

- –Fortran/C/C++
- –platform independent
- –parallel part is annotated by pragmas or directives
- –Users have to assure correctness of the parallelization
- –incremental parallelization
- –executed in serial if pragmas/directives are omitted

Available OpenMP compilers:

Omni(RWCP), KAI, PGI, SGI, etc.

Objectives

Evaluate Omni OpenMP compiler and commercial OpenMP compilers

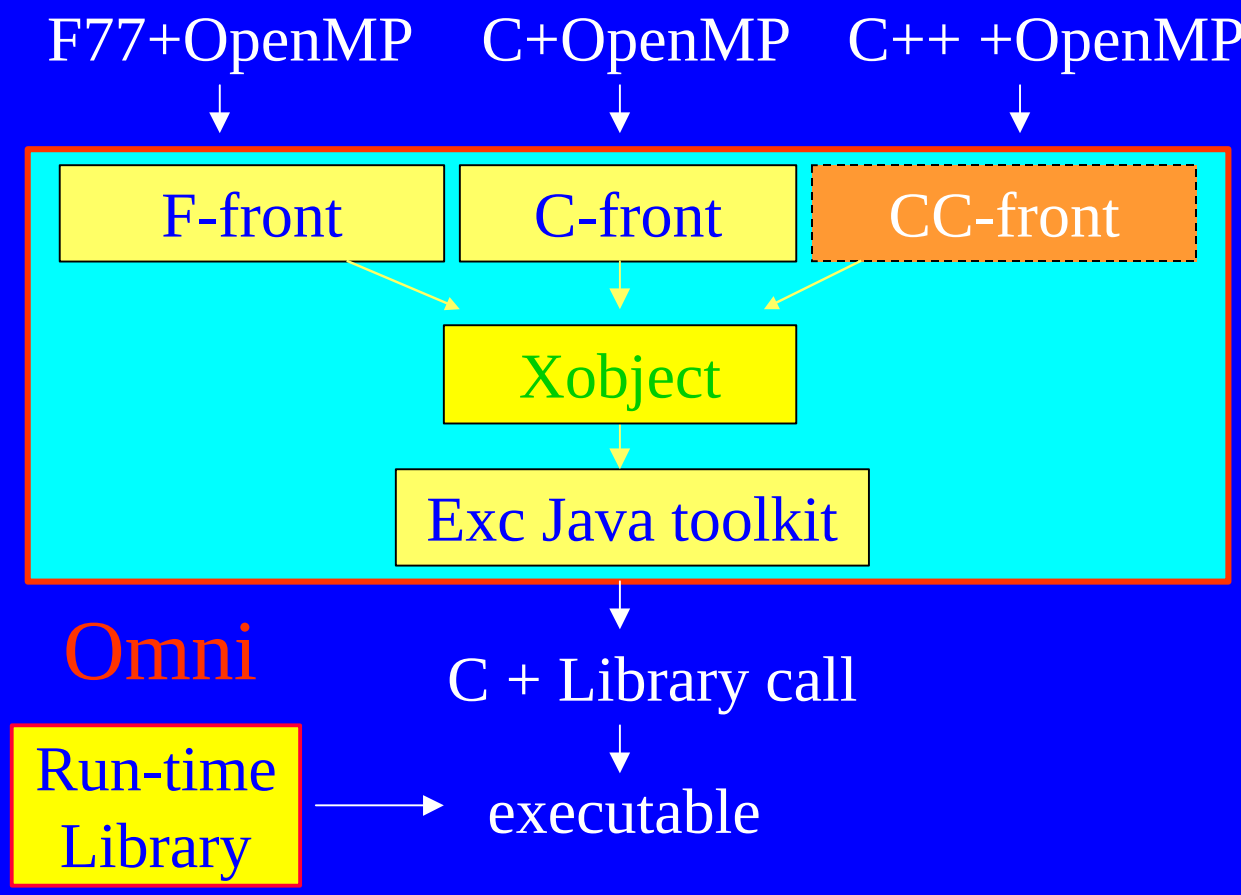
- measure each directive's overhead
  --> Microbench
- performance improvements of some example loops
  --> Parkbench

Show the characterization of the machine

Contents

- Omni overview
- Evaluation environment
- Microbenchmarks
- Results
- Parkbench
- Results
- Summary

Omni OpenMP Compiler



Omni overview

- Open source and Free
- Input: C or FORTRAN77 with OpenMP
- Output: parallel C with run-time library calls
- Front-end, Exc Java toolkit, and Run-time library
- POSIX thread, Solaris thread or StackThreads
- StackThreads(U. Tokyo) support nested parallel
- threads can be bind to the CPU on Solaris
- cluster enabled OpenMP on SCASH(beta state)

*Omni Ver. 1.2 is available at the following site:

http://pdplab.trc.rwcp.or.jp/pdperf/Omni/

CD-ROM will be available at SC2000@Dallas

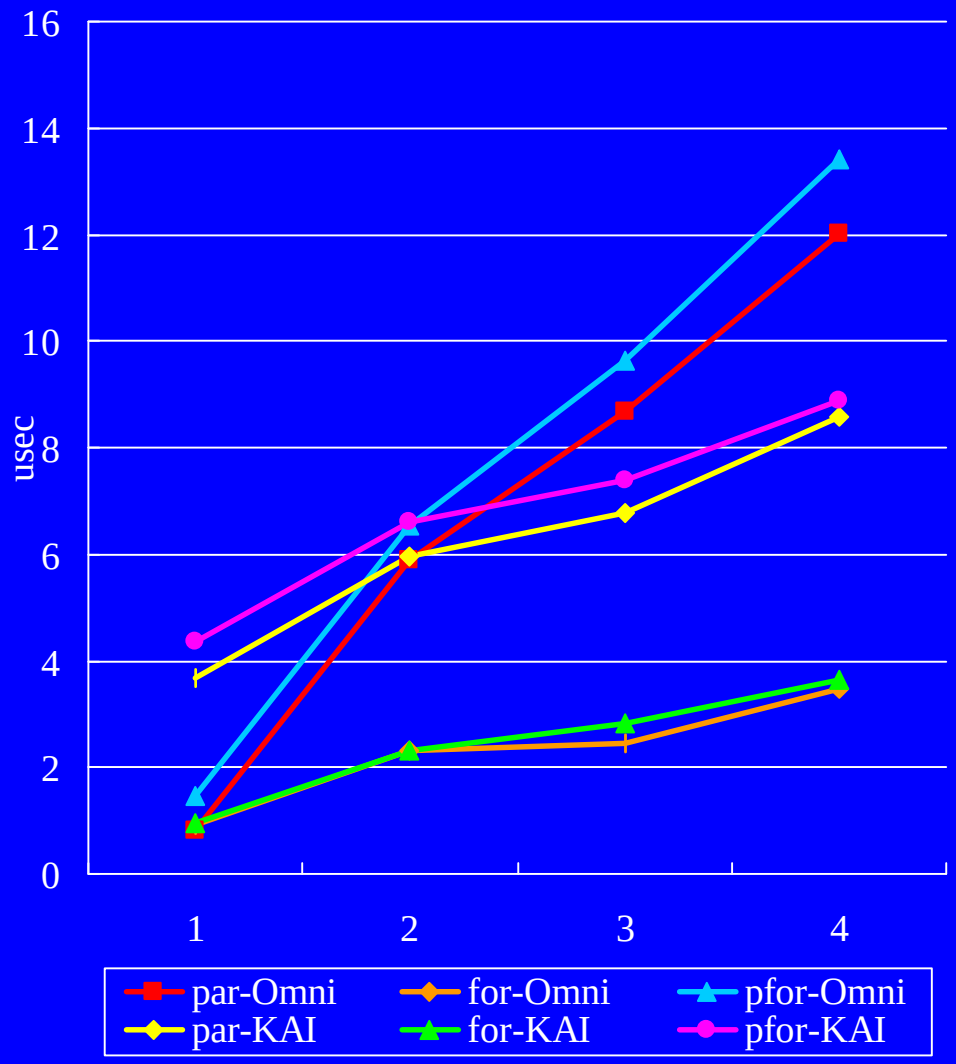
Evaluation environment

- SUN Enterprise450(Ultra sparc 300MHz x4, 2MB cache)
  Solaris2.6, SUNWspro4.2 C compiler
  - –KAP/Pro Toolset C 3.9
  - –Omni 1.2(Solaris thread + spin wait)
- COMPAQ ProLiant6500(P-II Xeon 450MHz x4, 1MB cache)
  RedHat6.0+2.2.12, egcs-1.1.2(Omni only)
  - –PGI pgcc 3.1-2
  - –Omni 1.2(spin wait)

Microbench

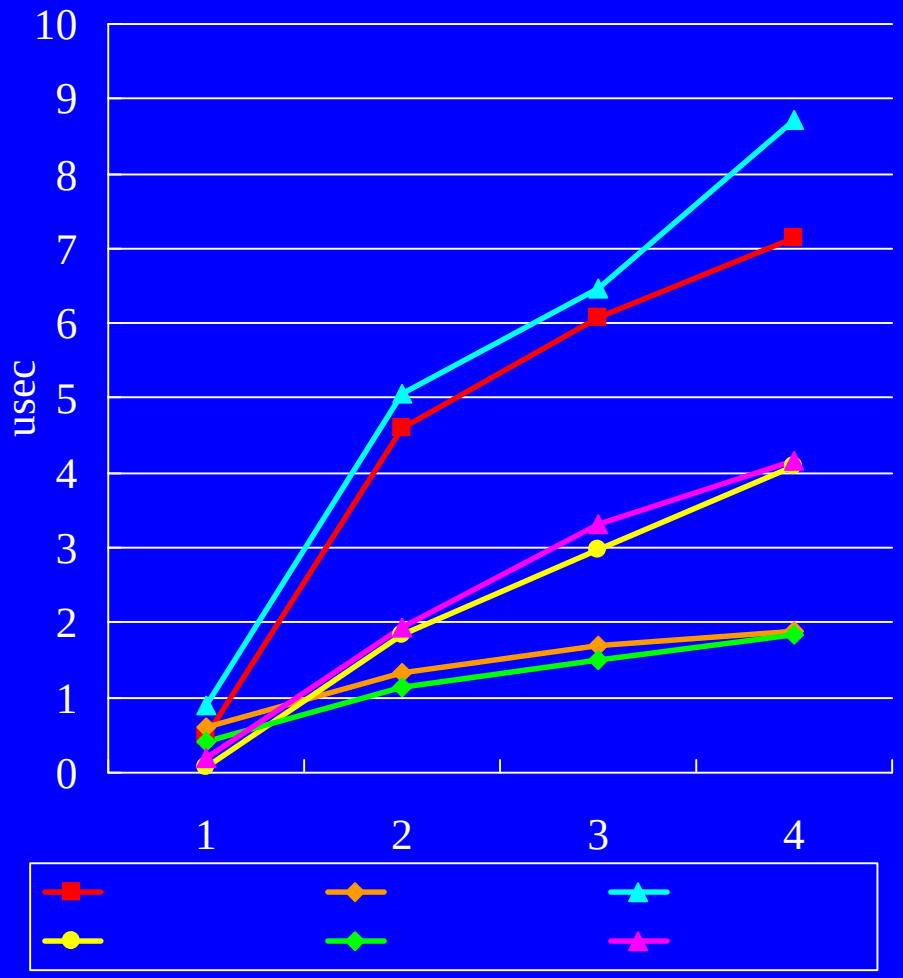
- OpenMP benchmark developed at Edingburgh Univ.
- C and Fortran versions are available
- Microbench measures:
  - [1] ten kinds of OpenMP directives overhead
  - [2] parallel loop scheduling overhead
- time difference between a program with and without OpenMP pragmas is the overhead of OpenMP

Overhead on SUN(Omni & KAI)



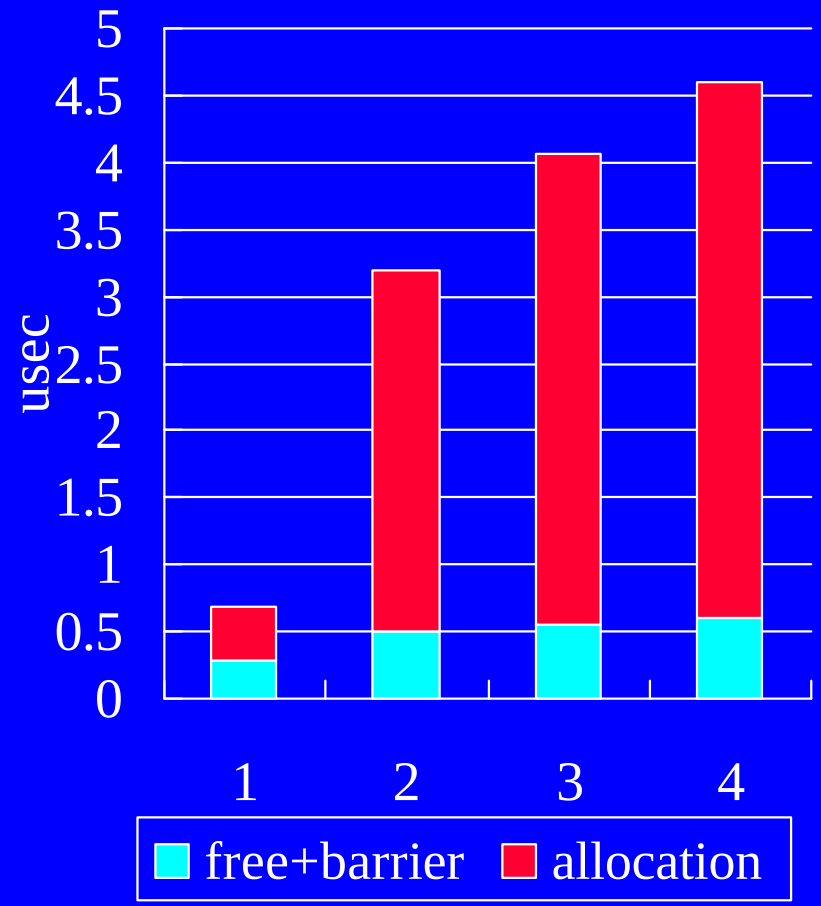
- Omni achieves comparable performance to KAI: ‘for’
- Omni overhead of ‘parallel’ and ‘parallel-for’ scales poorly
- ‘for’ is faster than ‘parallel’ and ‘parallel-for’ more than twice

Overhead on COMPAQ (Omni & PGI)



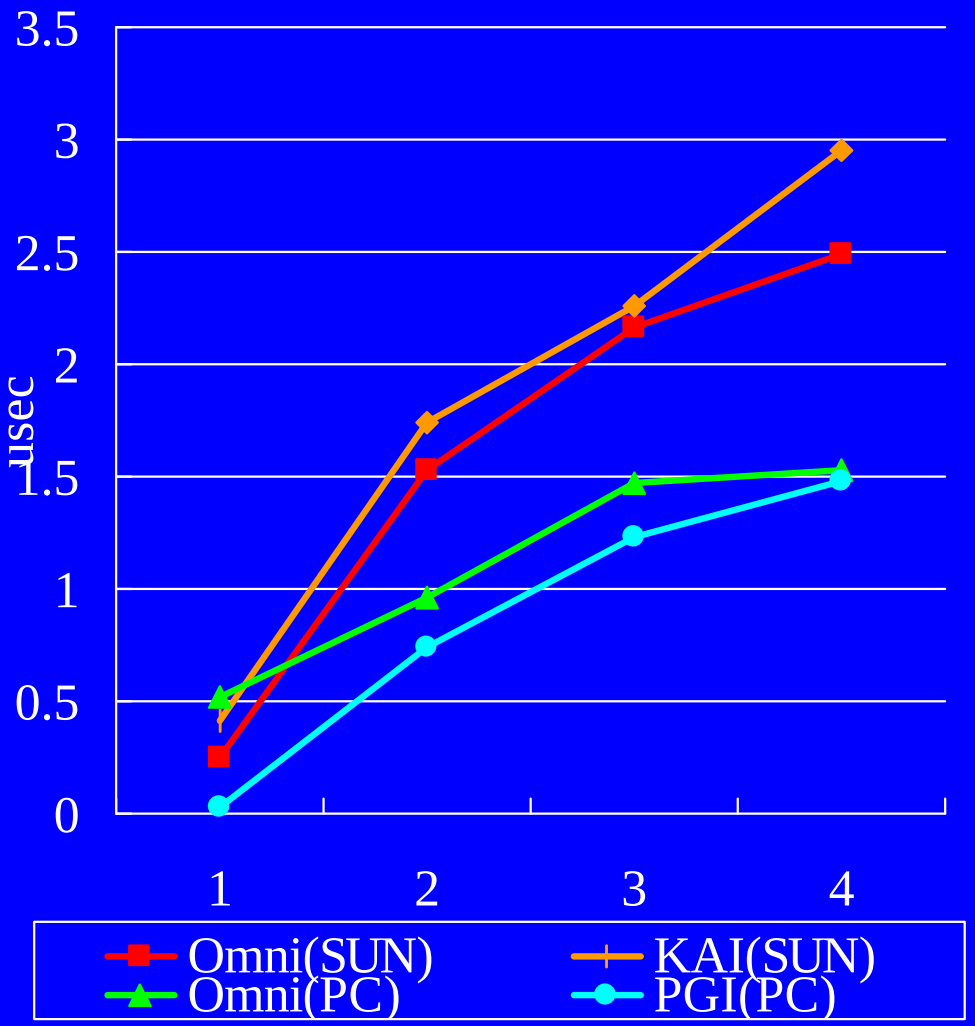
- PGI shows low overhead
- Omni takes more overhead for parallel and parallel-for

Omni ‘parallel’ overhead breakdown



- Thread allocation is once
- free thread is managed by the linked list in Omni lib.
- list is operated exclusively
- thread allocation from free-list is slow

barrier overhead



- Omni achieves comparable performance to commercial OpenMP compilers

Parkbench

–LowLevel/Kernel/Application/NPB
–MPI or PVM is used for parallel execution
–LowLevel benchmark measures HW parameter
–'rinf1' measures execution time of a set of simple loops
–it calculates the asymptotic performance rate in Mflop/s, R-infinity(r_∞), and the loop length which required to achieve performance of one half of R-infinity, N-half($n_{1/2}$)

$$l = q * \frac{n + n_{1/2}}{r_\infty}$$

n: vector length which achieve R-infinity
l: time, q: vector operations

$$r = \frac{q * n}{l} = \frac{r_\infty}{\left(1 + n_{1/2} / n\right)}$$

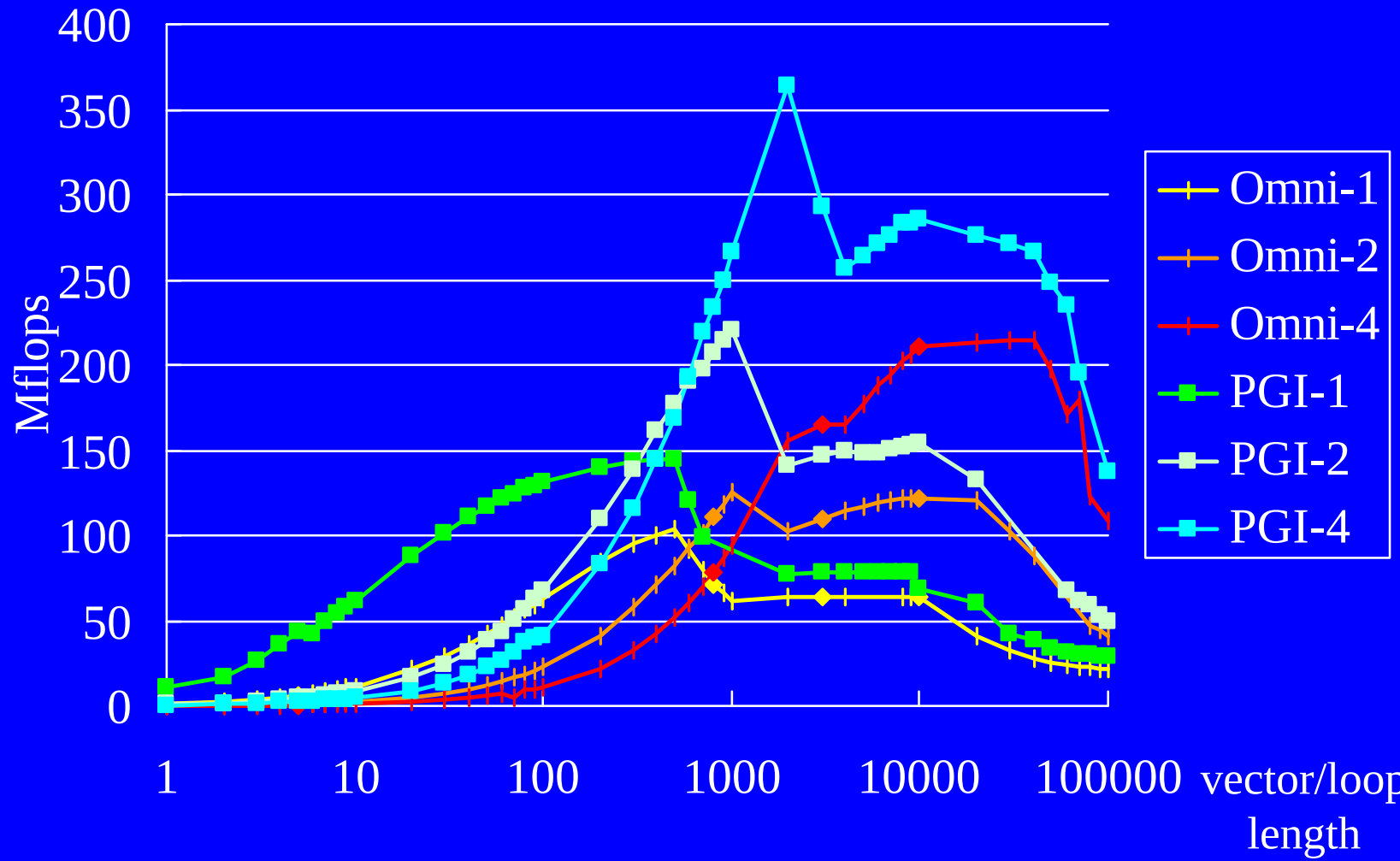
r: performance rate

Rinf1

–rinf1 contains 17 simple kernel loops
–parallelize ‘rinf1’ using OpenMP
–measure rinf1 performance(1/2/4 PE)
    loop(vector) length: 100,000
–the results of kernel 3, 6, and 16 are shown
–graph shows Rn that is calculated using execution time, not R-infinity

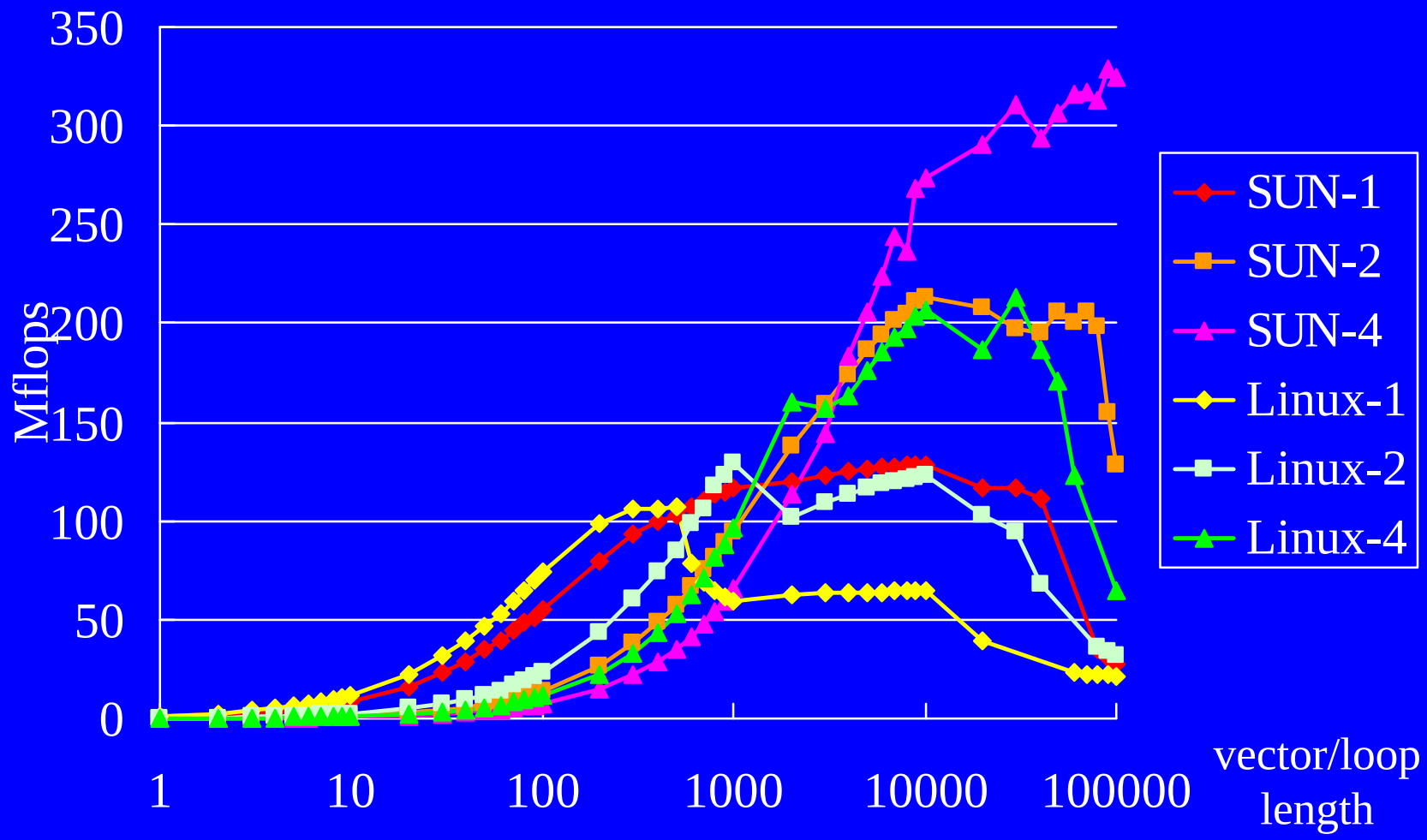
Kernel-16 on COMPAQ

A[i]=S*B[i]+C[i]

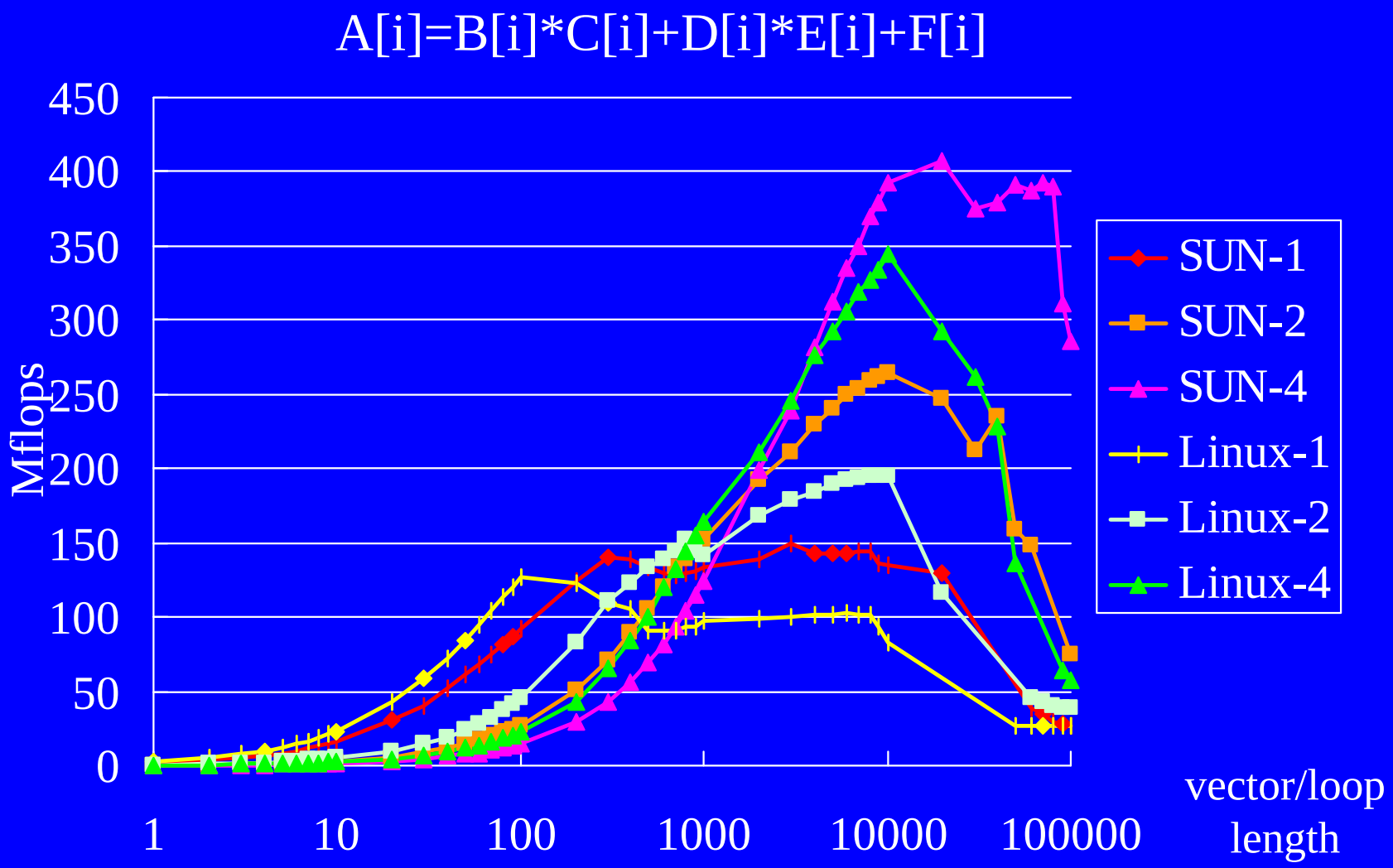


Omni Kernel-3(SUN vs COMPAQ)

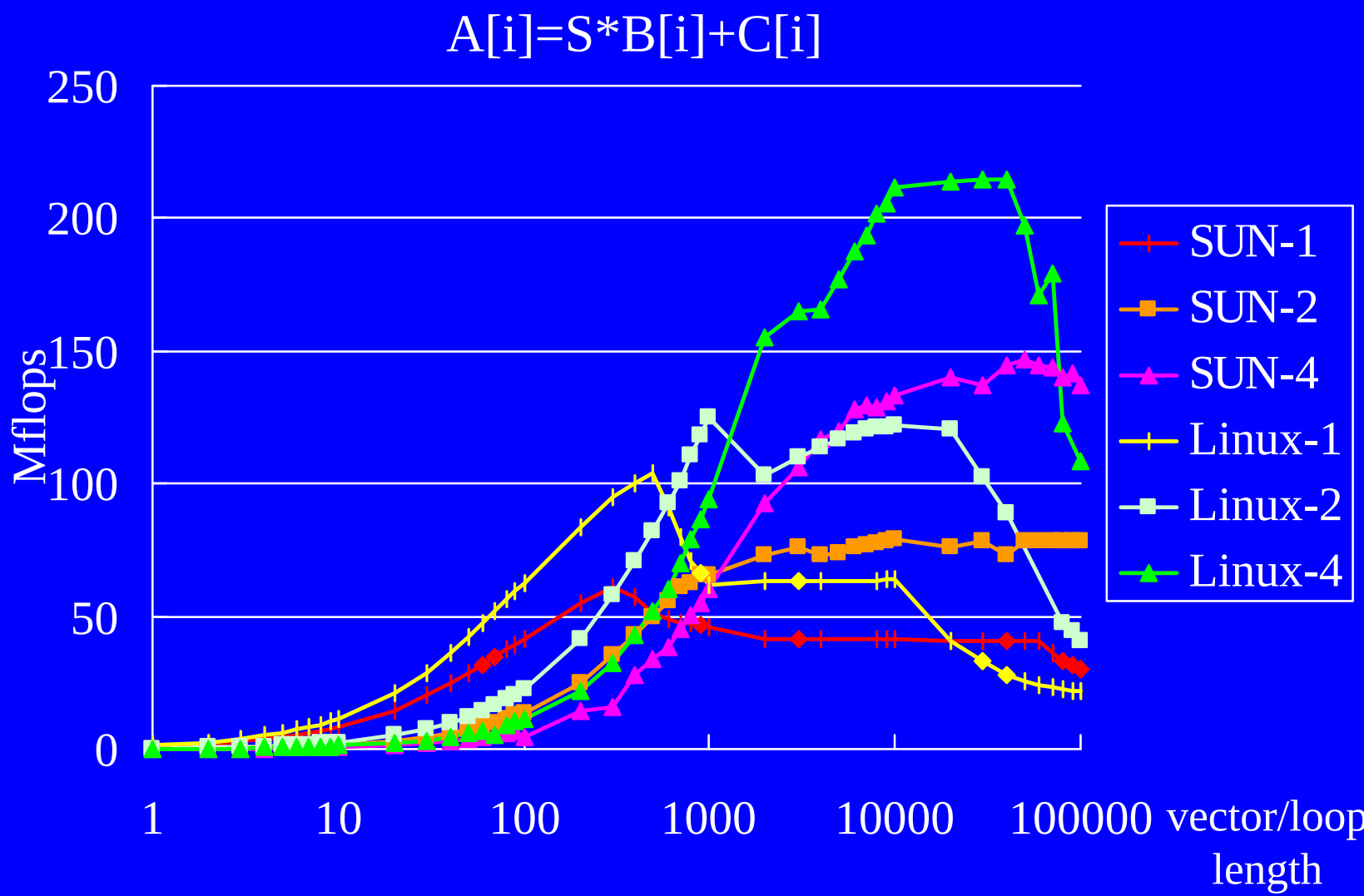
A[i]=B[i]*C[i]+D[i]



Omni Kernel-6(SUN vs COMPAQ)



Omni Kernel-16(SUN vs COMPAQ)



Summary

–performance evaluation of OpenMP compilers, Omni, PGI and KAI, are presented

–Omni, an open source OpenMP compiler, achieves comparable performance to the KAI on SUN

–PGI shows good performance, especially short vector length

–OpenMP is effective to improve performance if the loop has enough length