

OMPI: A portable C compiler for OpenMP V2.0

University of Ioannina

Elias Leontiadis

George Tzoumas

Vassilios V. Dimakopoulos



Presentation

- **Introduction**
- OMPI
- OMPI Performance
- Conclusions

The OpenMP specification

- High level API for parallel programming in a shared memory environment

- Fortran
 - Version 1.0, October 1997
 - Version 1.1, November 1999
 - Version 2.0, November 2000

- C/C++
 - Version 1.0, October 1998
 - Version 2.0, March 2002
 - New features such as
 - timing routines
 - *copyprivate* and *num_threads* clauses
 - variable reprivatization
 - static threadprivate

OpenMP compilers

- Commercial compilers for specific machines
 - SUN, SGI, Intel, Fujitsu, etc.
- OpenMP compiler projects (usually portable)
 - Nanos
 - OdinMP/CCp
 - Intone project
 - Omni



Presentation

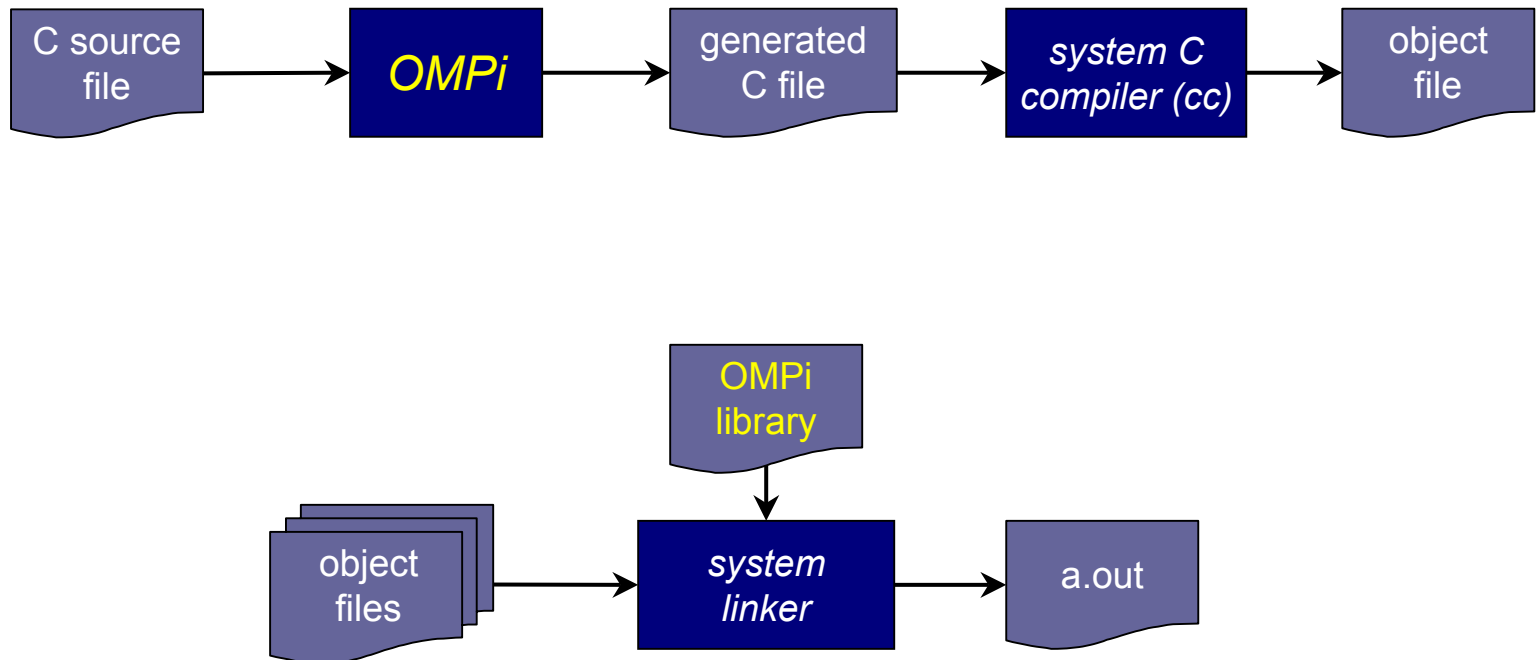
- Introduction
- **OMPI**
- OMPI Performance
- Conclusions



OMPI

- Portable C compiler for OpenMP
- Adheres to V.2.0
- Produces ANSI C code with POSIX threads library calls
- Written entirely in C

Compilation process



Code transformations

■ *parallel* construct

- code is moved into a (thread) function
 - a struct is declared containing pointers to non-global *shared* variables
 - *private* variables are redeclared locally in the function body
- original code is replaced by code that creates a team of threads executing the function
- master thread executes the function, too

Example

```
int a;    /* global */
int main()
{
    int b, c;

    #pragma omp parallel num_threads(3) \
        private(c)
    {
        c = b + a;
        . . .
    }
}
```

```
int a;

typedef struct { /* shared vars structure */
    int (*b); /* b is shared, non-global */
} par0_t;

int main()
{
    int b, c;
    _omp_initialize();
    {
        /* declare par0_vars, the shared var struct */
        _OMP_PARALLEL_DECL_VARSTRUCT(par0);
        /* par0_vars->b will point to real b */
        _OMP_PARALLEL_INIT_VAR(par0, b);
        /* Run the threads */
        _omp_create_team(3, _OMP_THREAD, par0_thread,
            (void *) &par0_vars);
        _omp_destroy_team(_OMP_THREAD->parent);
    }
}

void *par0_thread(void *_omp_thread_data)
{
    int _dummy = _omp_assign_key(_omp_thread_data);
    int (*b) = &_OMP_VARREF(par0, b);
    int c;
    c = (*(b)) + a;
    . . .
}
```

Work sharing constructs

- ***sections*** construct

- a **switch-case** block is created
- the code of each ***section*** is moved into a case of the switch block
- any thread may execute any section

- ***for*** construct

- each thread computes the bounds of the next chunk to execute
- then, if a chunk is available, executes the for-loop within the computed bounds

Threads

- a pool of threads is created when the program starts, all threads are sleeping
- initial pool size is number of CPUs or `$OMP_NUM_THREADS`
- user can request a specific number of threads by using the *num_threads* clause or *omp_set_num_threads()*



Presentation

- Introduction
- OMPI
- **OMPI Performance**
- Conclusions

Benchmarks

- NAS parallel benchmarks
 - OpenMP C version of ported by Omni group (v2.3)
 - Results for Class W
- Edinburgh University microbenchmarks (EPCC)
 - Measure synchronization overheads

Platforms

- SGI origin 2000 system
 - 48 MIPS R10000 CPUs
 - IRIX 6.5
- Compaq proliant ML 570
 - 2 Intel Xeon CPUs
 - Redhat Linux 9.0
- SUN E-1000 Server
 - 4 Sparc CPUs
 - Solaris 5.7



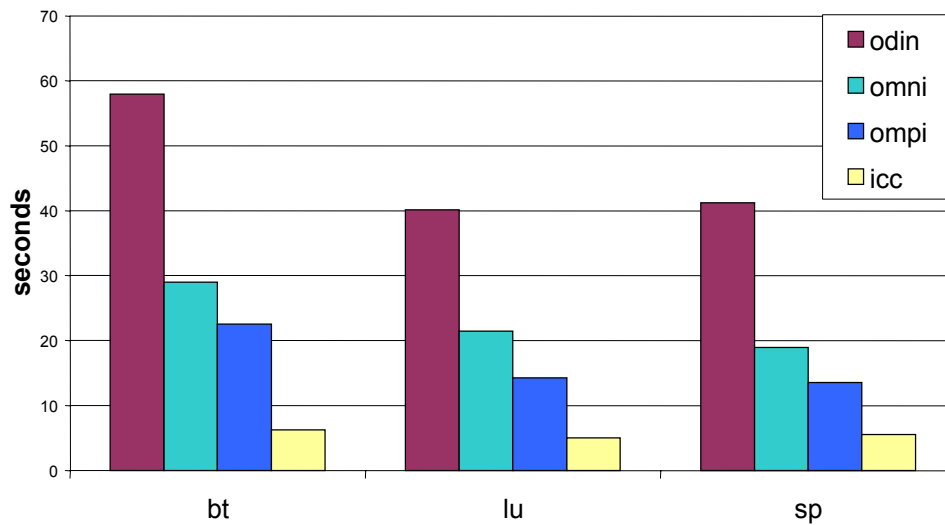
Compilers

- OdinMP/CCp v1.02
- Omni v1.4a
- Intel C/C++ compiler (ICC) v7.1
- Mipspro v7.3

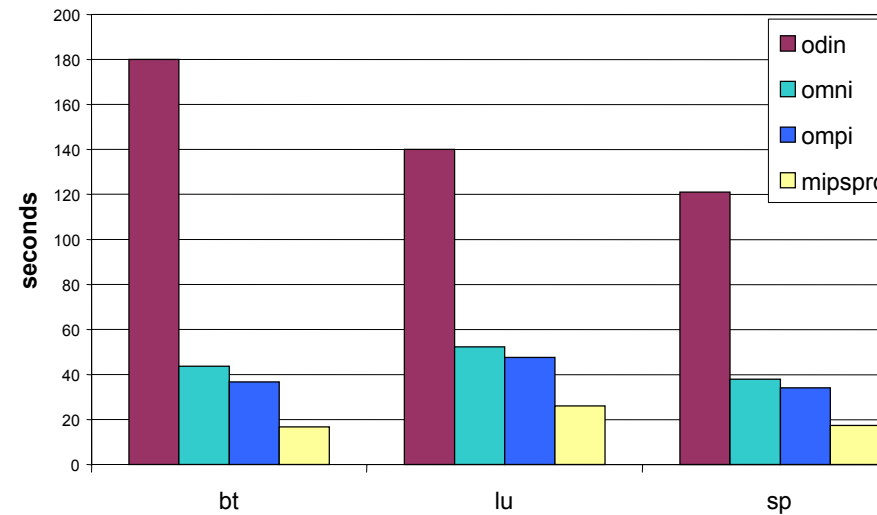
NAS parallel benchmarks

Compilation Time

Compilation times for 2-CPU Linux system



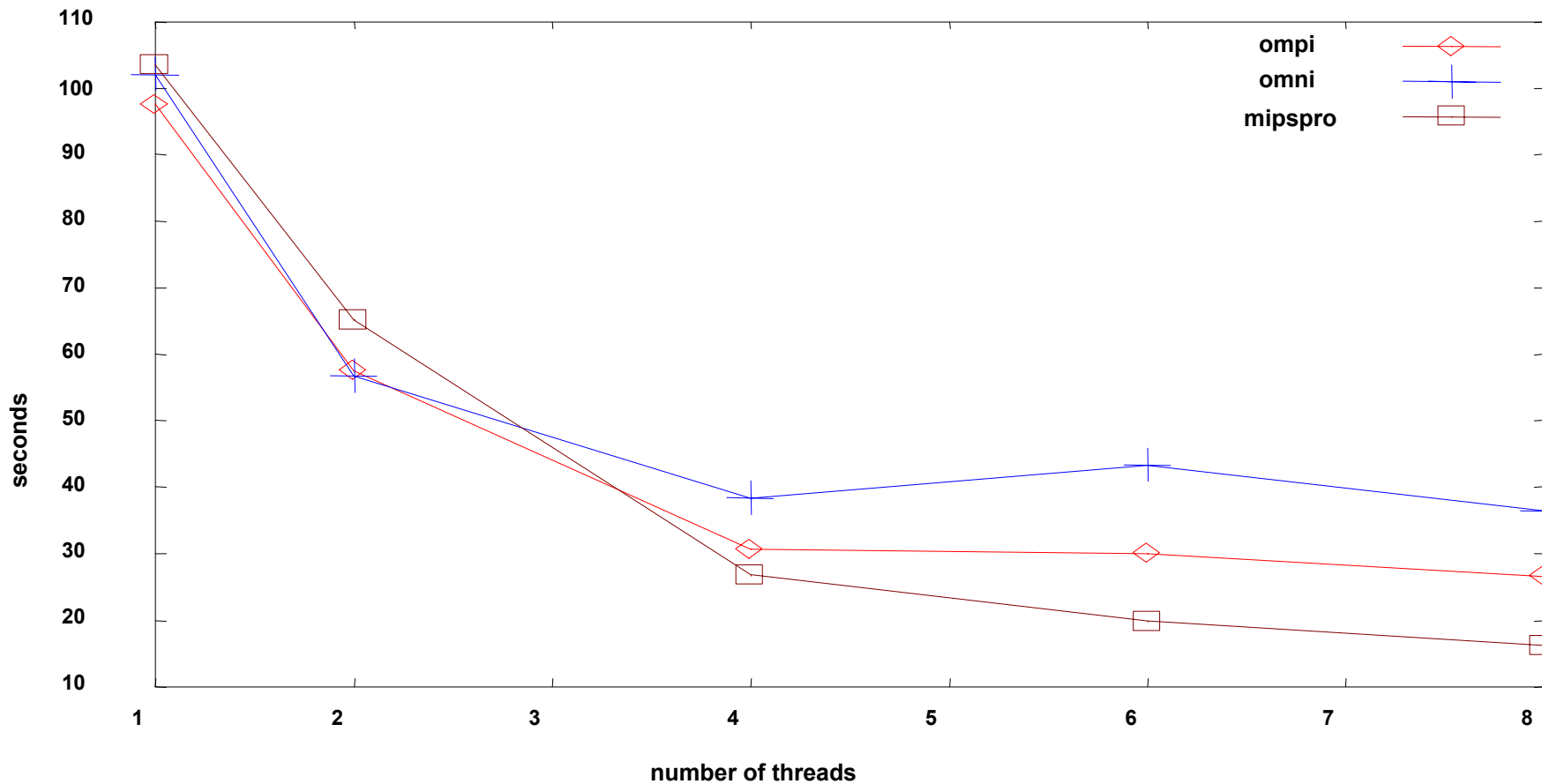
Compilation times for the SGI Origin 2000 system



NAS parallel benchmarks

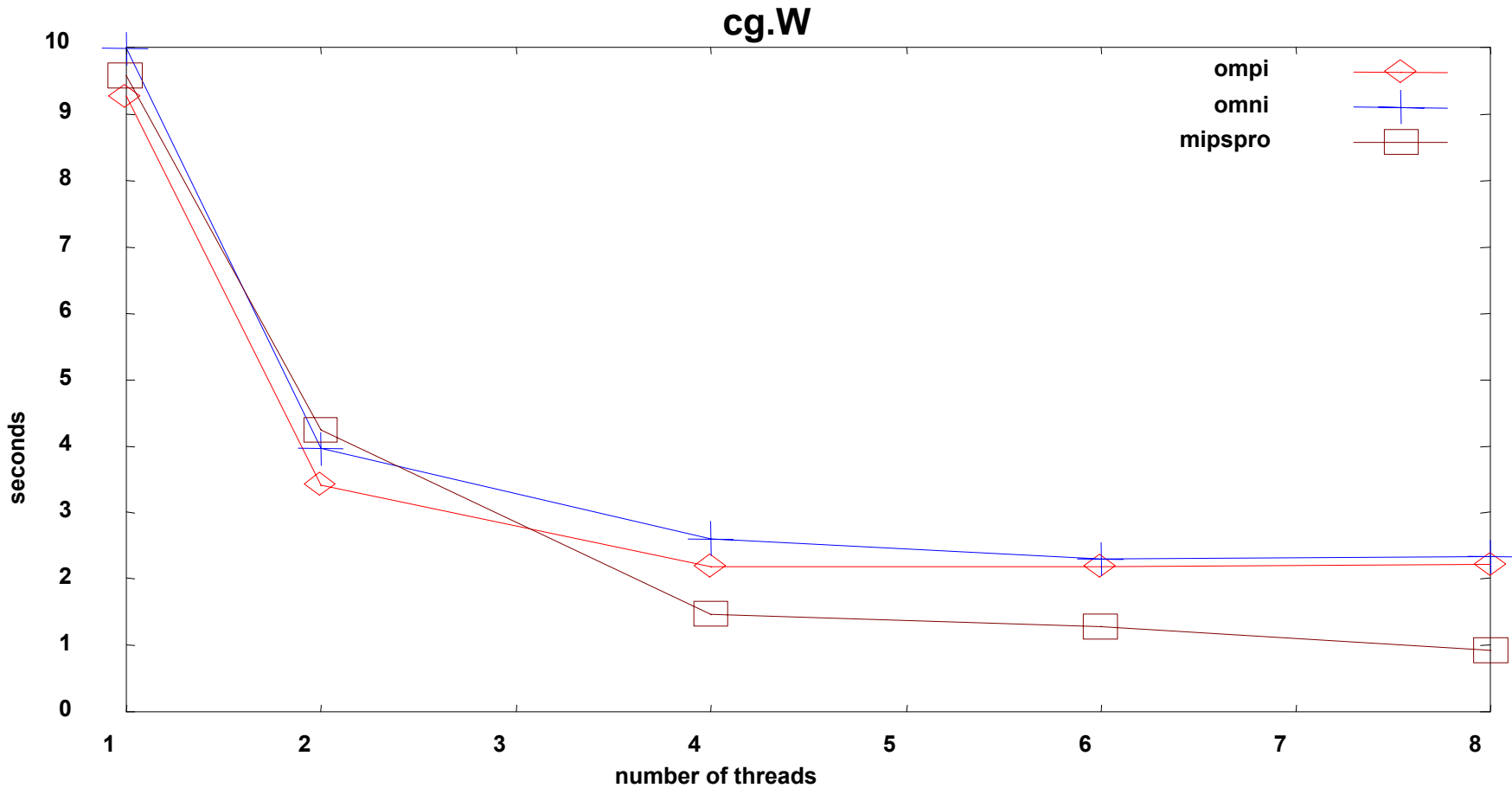
SGI Origin 2000 (execution time)

bt.W



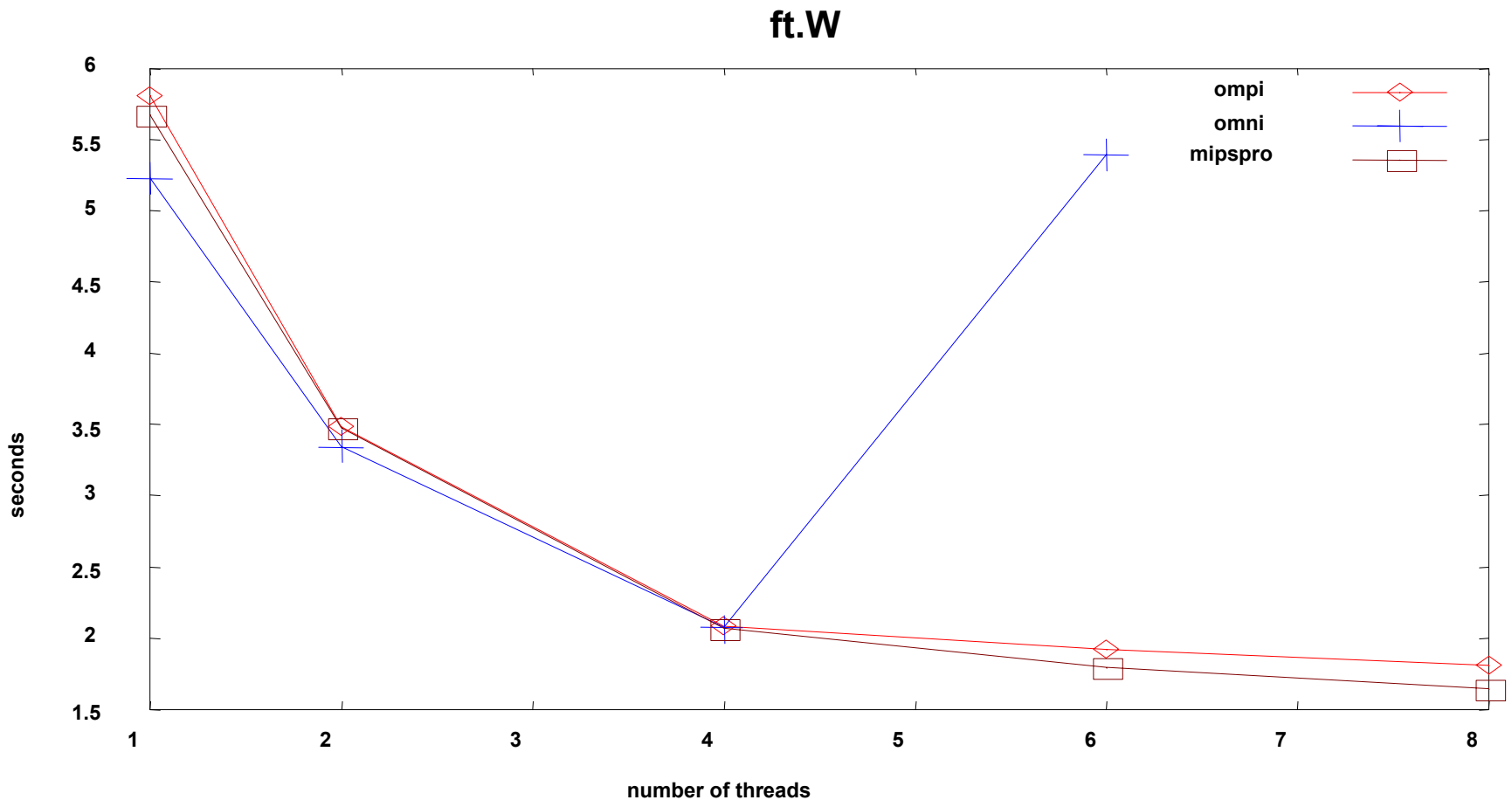
NAS parallel benchmarks

SGI Origin 2000



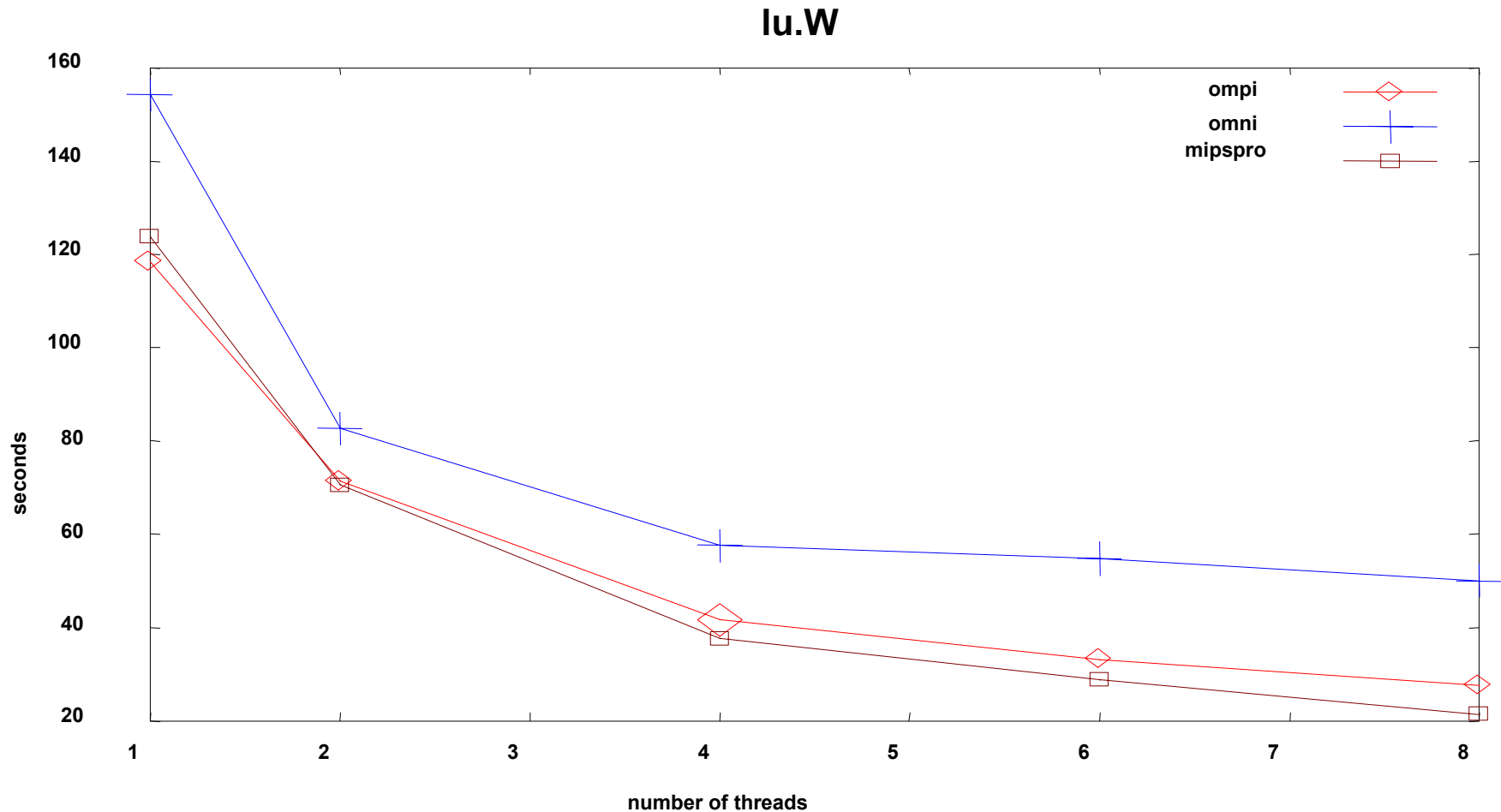
NAS parallel benchmarks

SGI Origin 2000



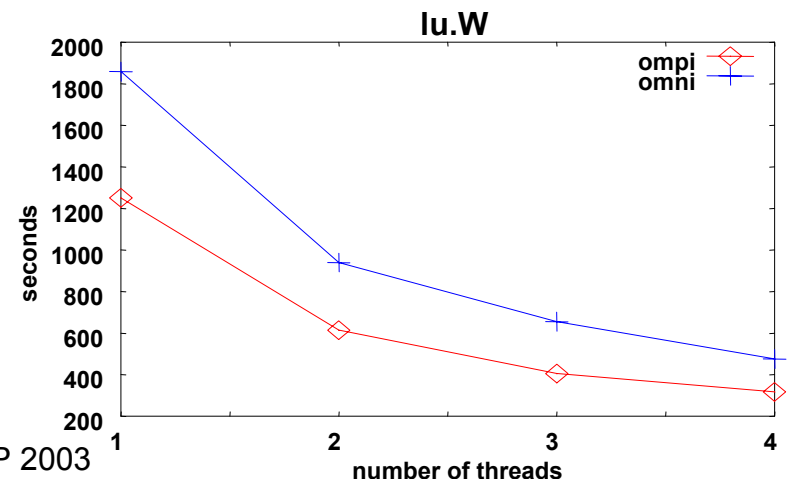
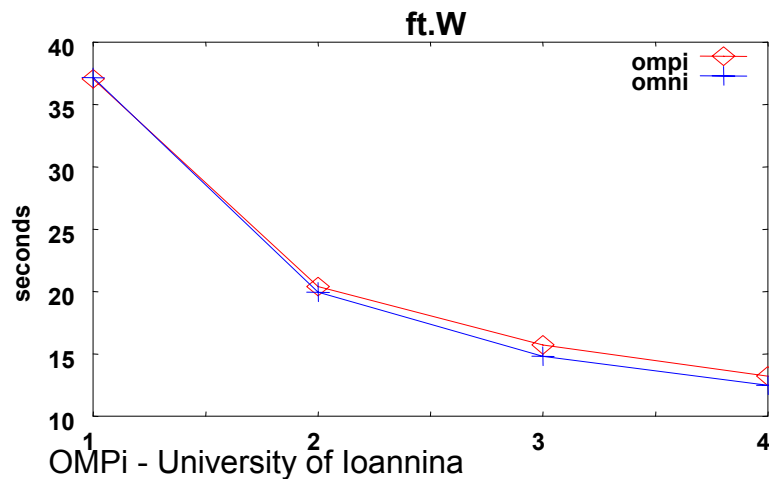
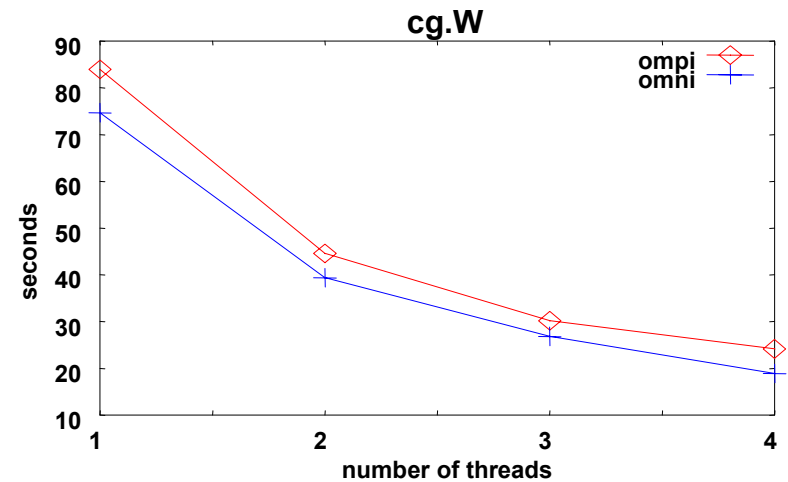
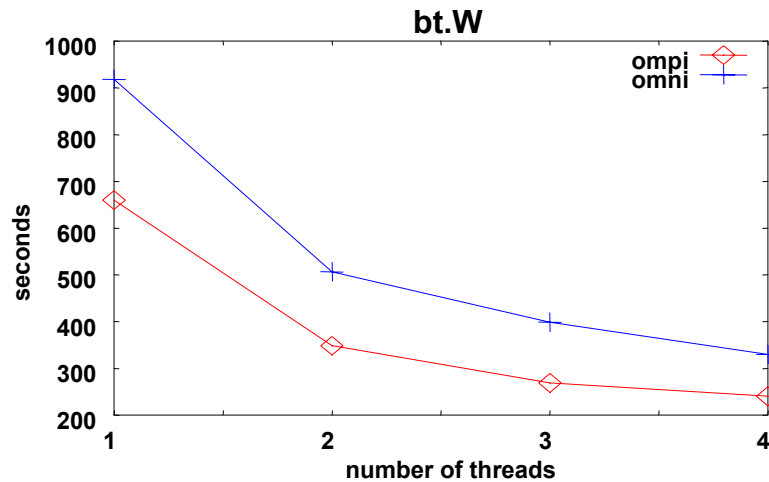
NAS parallel benchmarks

SGI Origin 2000



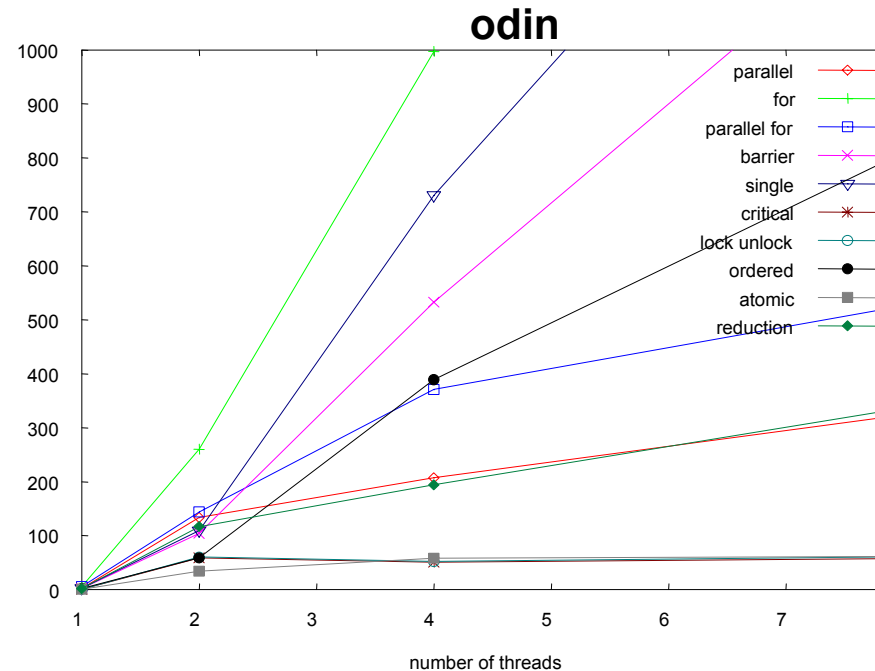
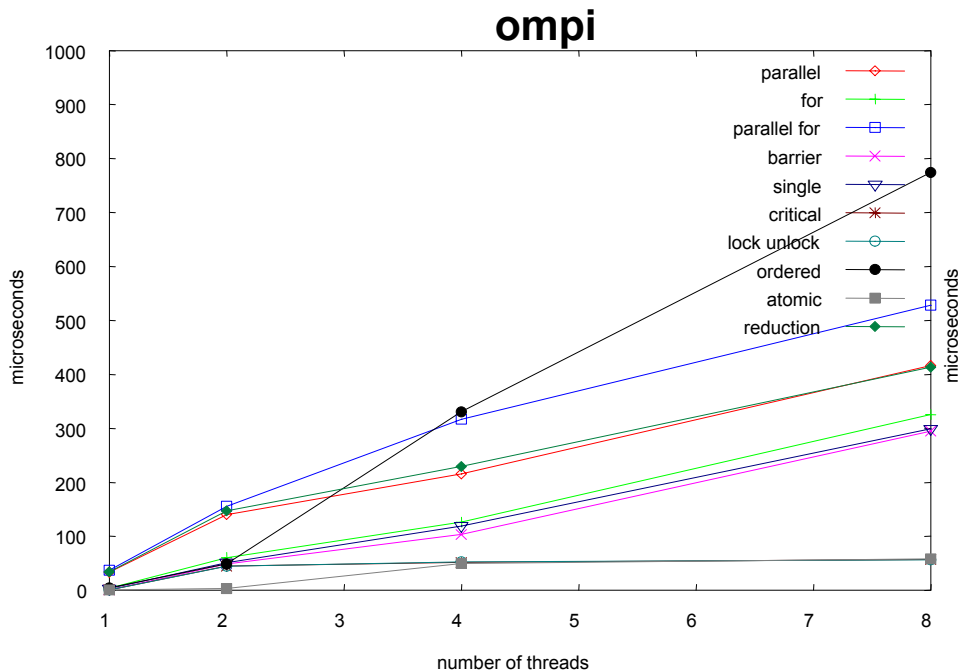
NAS parallel benchmarks

Sun E-1000



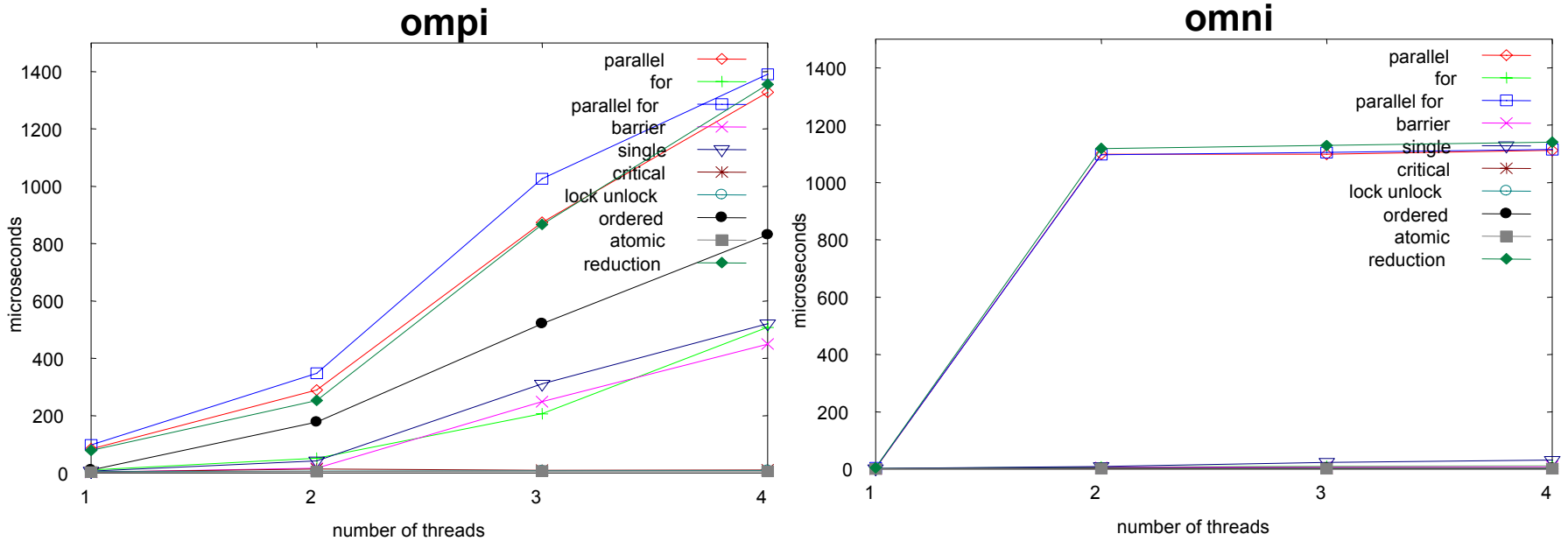
EPCC microbenchmarks

SGI (overheads)



EPCC microbenchmarks

SUN





Presentation

- Introduction
- OMPI
- OMPI Performance
- **Conclusions**



Conclusions

- C compiler for OpenMP V.2.0
- Written in C, generated code uses pthreads
- Tested on Linux, Solaris, Irix
- Performance satisfactory, comparable with native compilers



Current status

- Target solaris threads, sproc
- Improve overheads (e.g. ordered)
- Improve produced code (optimizations)
- Profiling code



Thank you

<http://www.cs.uoi.gr/~ompi>