

Portable Support and Exploitation of Nested Parallelism in OpenMP

Panagiotis E. Hadjidoukas Laurent Amsaleg

IRISA/INRIA, Rennes, France

TEXMEX group

{phadjido,lamsaleg}@irisa.fr

EWOMP 2004

Stockholm, Sweden, 18-21 Oct. 2004



Introduction

- OpenMP and Nested Parallelism
- Most implementations serialize inner levels
- The decision is taken by the runtime library
- Support for nested parallelism
 - Intel
 - Fujitsu PRIMEPOWER
 - Native Omni and Omni/ST
 - NANOS
- NANOS runtime library (NthLib)

Contributions

- Portable NthLib
- Integration of NthLib into third-party OpenMP compilers
- Portable runtime support for nested parallelism
- Experimental implementation of the workqueuing model
- Parallel data clustering algorithm

- Goal: General and Portable implementation of the NANOS execution environment
 - Compiler
 - Runtime Library
 - Multiprogramming module

NANOS Runtime Library (NthLib)

- Two primitives for spawning parallelism
 - Inner level: work descriptors
 - Higher levels: nanothreads
 - Optimizations focus on work descriptors
- Alternative implementation of NthLib
 - Portability (POSIX API)
 - Virtual processors: POSIX Threads
 - User-level threads: setjmp/longjmp or ucontext
 - Synchronization: POSIX mutexes or spinlocks
 - Lazy stack allocation policy
 - Both descriptors are the same
 - No special memory locations for work descriptors
 - Minimal memory consumption
 - Easier management of parallelism on SDSM

- Both descriptors and stacks are recycled
- Stack handoff
 - Peer-to-peer scheduling of nanothreads
 - A finished nanothread
 - Picks the next ready descriptor
 - Associates its user-level thread with that descriptor
 - Restarts the user-level thread

OpenMP runtime support

- Integration of NthLib into OpenMP implementations
 - Omni, OMPI, ORC, Intone, NANOS
 - Porting of the runtime libraries
- Extensions in NthLib
 - (Nanthread) descriptor
 - General implementation of nanothread barriers
 - Arbitrary number of threads / Multiple levels
 - Busy waiting not applicable
- Omni 1.6
 - Portability (libompc)
 - Efficient Support for Nested Parallelism (libompst)
- OMPI 0.8.1
 - Written in C, OpenMP v. 2.0
 - Extensive use of the POSIX Threads API

Management of Nested Parallelism

- Efficient support for many (OpenMP) threads and multiple levels of parallelism
 - Lightweight threading
 - Stack allocation policy
- Variation of the all-to-all scheme
 - First level: distribute descriptors to processors
 - Inner levels: insert in the local queue
- Other approaches
 - Omni/ST (pure all-to-all, random stealing, not portable)
 - NANOS Groups (inter-group stealing not allowed)
 - Groups can define the order idle processors visit queues

Workqueuing model

- Inherent runtime support (queues, descriptors)

```
#pragma omp parallel taskq
{
    #pragma omp task
    { work(); }
}
```

```
self = nth_self();
nth_depadd(self, 1);
{
    { nth_depadd(self, 1);
      nth = nth_create_ls(work,0,self,0);
      nth_to_lrq_end(self->vp_id, nth);
    }
}
nth_block();
```

Parallel data clustering algorithm

- Data clustering
 - Data analysis and data mining
- CURE (Clustering Using Representatives)
 - Hierarchical Algorithm
 - Each record is a cluster
 - Find and merge closest pair, Update distances
 - Repeat until K clusters have been created
- Features
 - Computational intensive [$O(n^2 \log n)$]
 - Nested Loop level parallelism
 - Asymmetric and Non-deterministic
 - Data intensive application

Parallel CURE – Update phase

```
find_nearest_neighbor(int i, int *idx, double *dist
{
    min_dist = +∞, min_idx = -1;
    for (j = 0; j < i; j++) {
        - if (entry j has been invalidated) continue;
        - if ((dist = compute_dist (i, j)) < min_dist)
            { min_dist = dist; min_idx = j };
    }
    *idx = min_idx; *dist = min_dist;
}

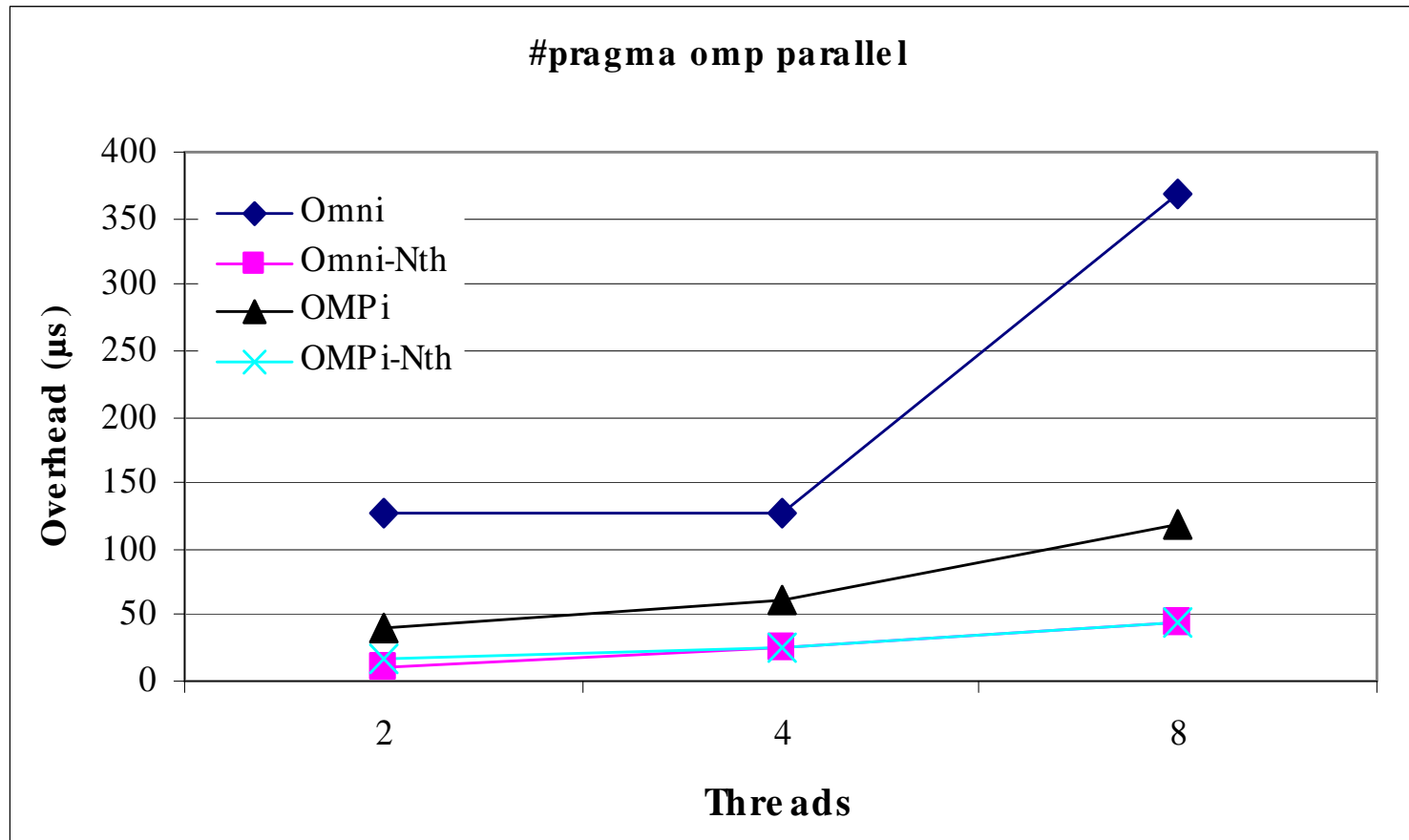
update_nearest_neighbors(int pair_low, int pair_high)
{
    for (i = pair_low+1; i < N; i++) {
        - if (entry i has been invalidated) continue;
        - if (entry i had neighbor pair_low or pair_high)
            find_nearest_neighbor(i, &nnb[i].index, &nnb[i].dist);
        - else if (pair_high < i)
            if ((dist = compute_dist(pair_low, i)) < nnb[i].dist)
                { nnb [i].index = pair_high; nnb[i].dist = dist; }
    }
}
```

Experiments

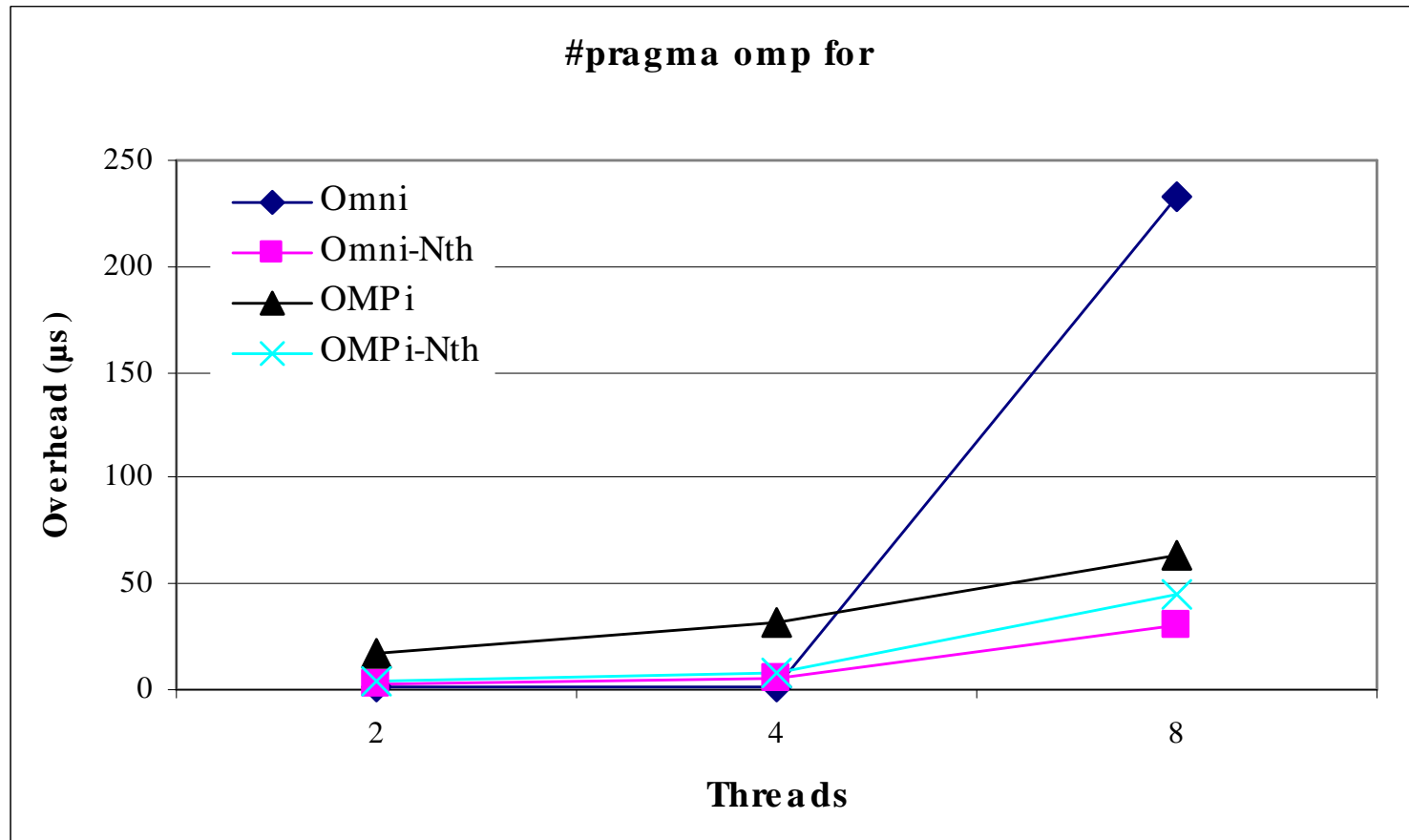
- Intel Pentium III system
- 4 processors (550MHz), 512 KB cache memory
- 1GB main memory

- Linux (2.6.6), NPTL
- Omni 1.6
- OMPi 0.8.1
- GNU gcc 3.3.2

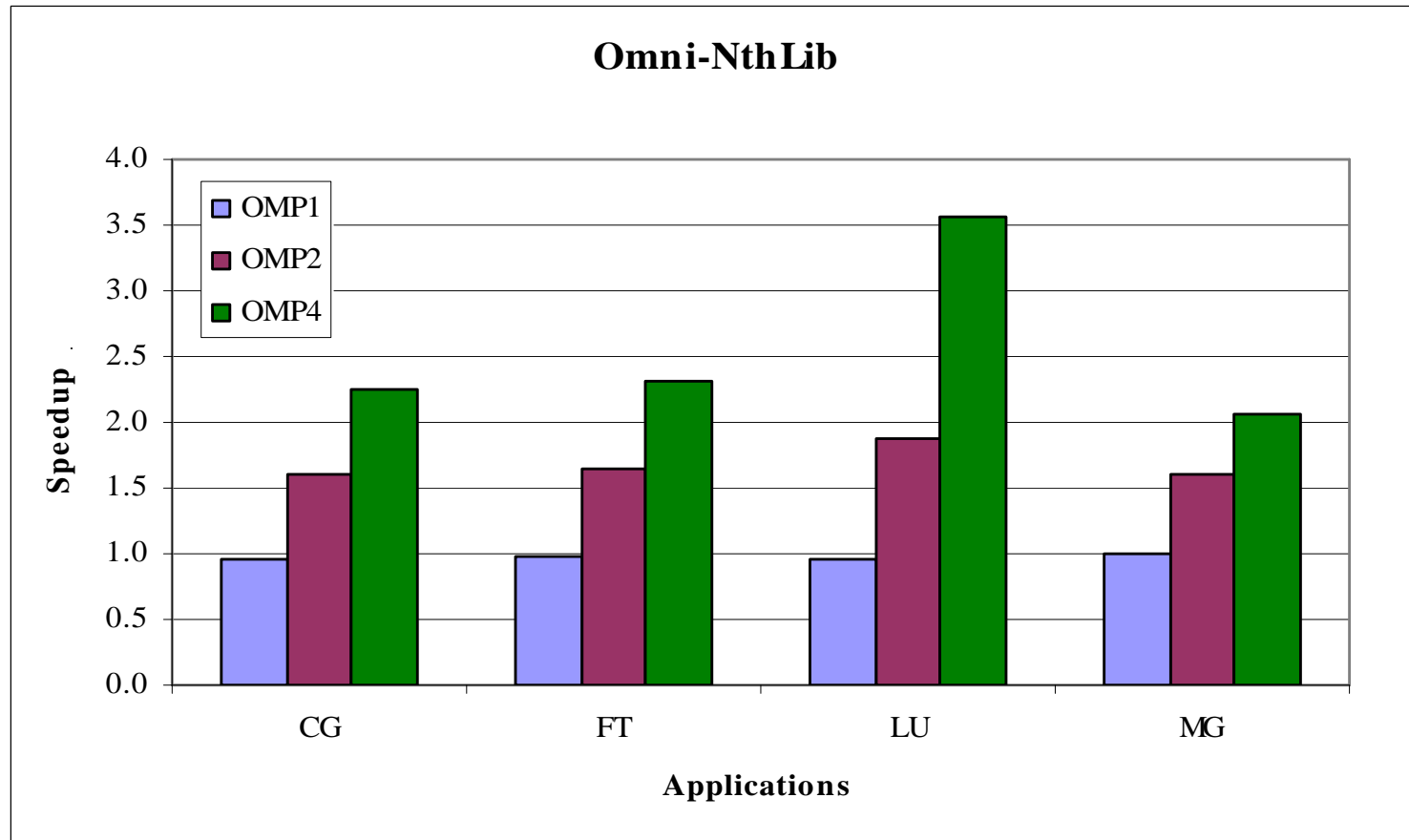
EPCC microbenmarks (parallel)



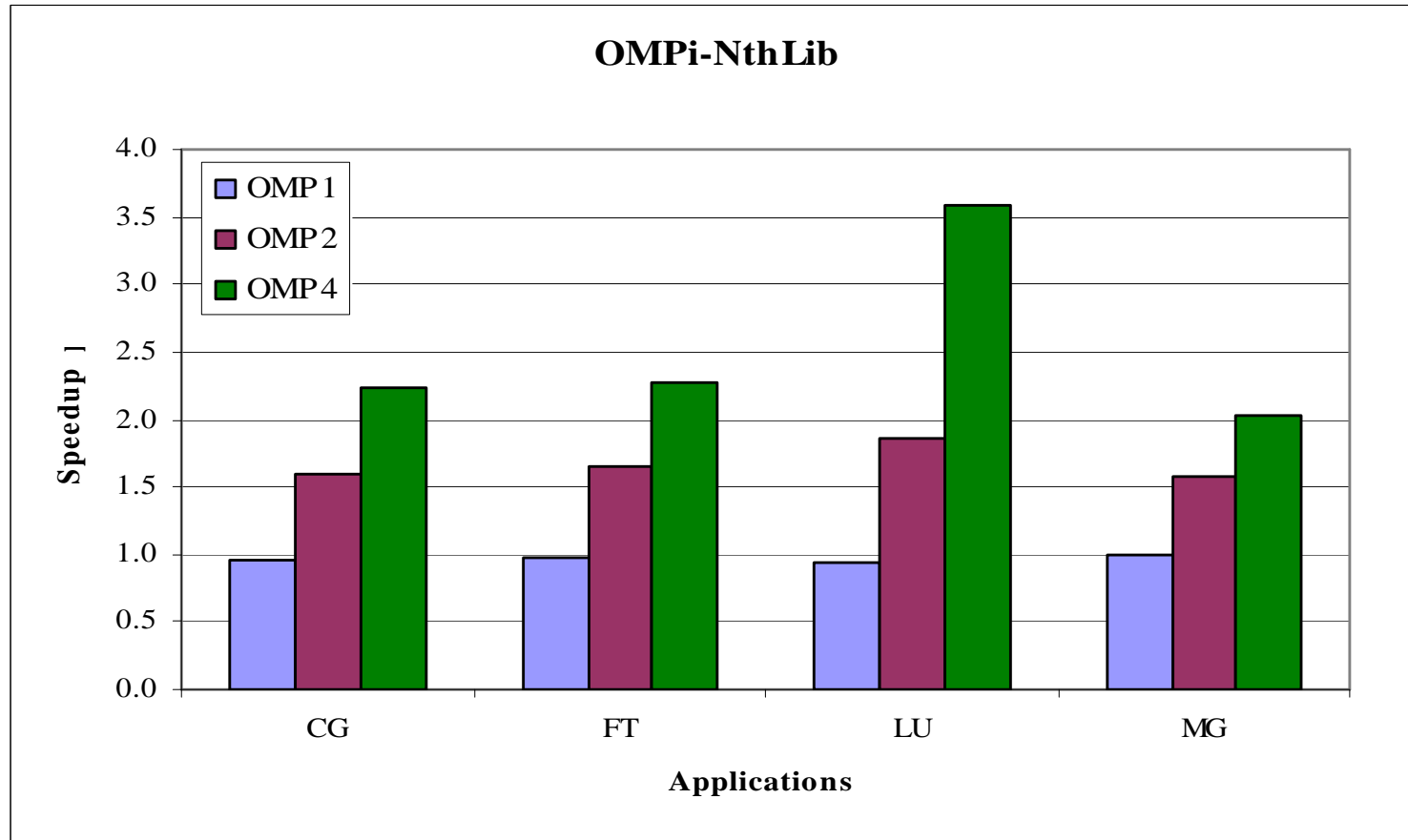
EPCC microbenmarks (for)



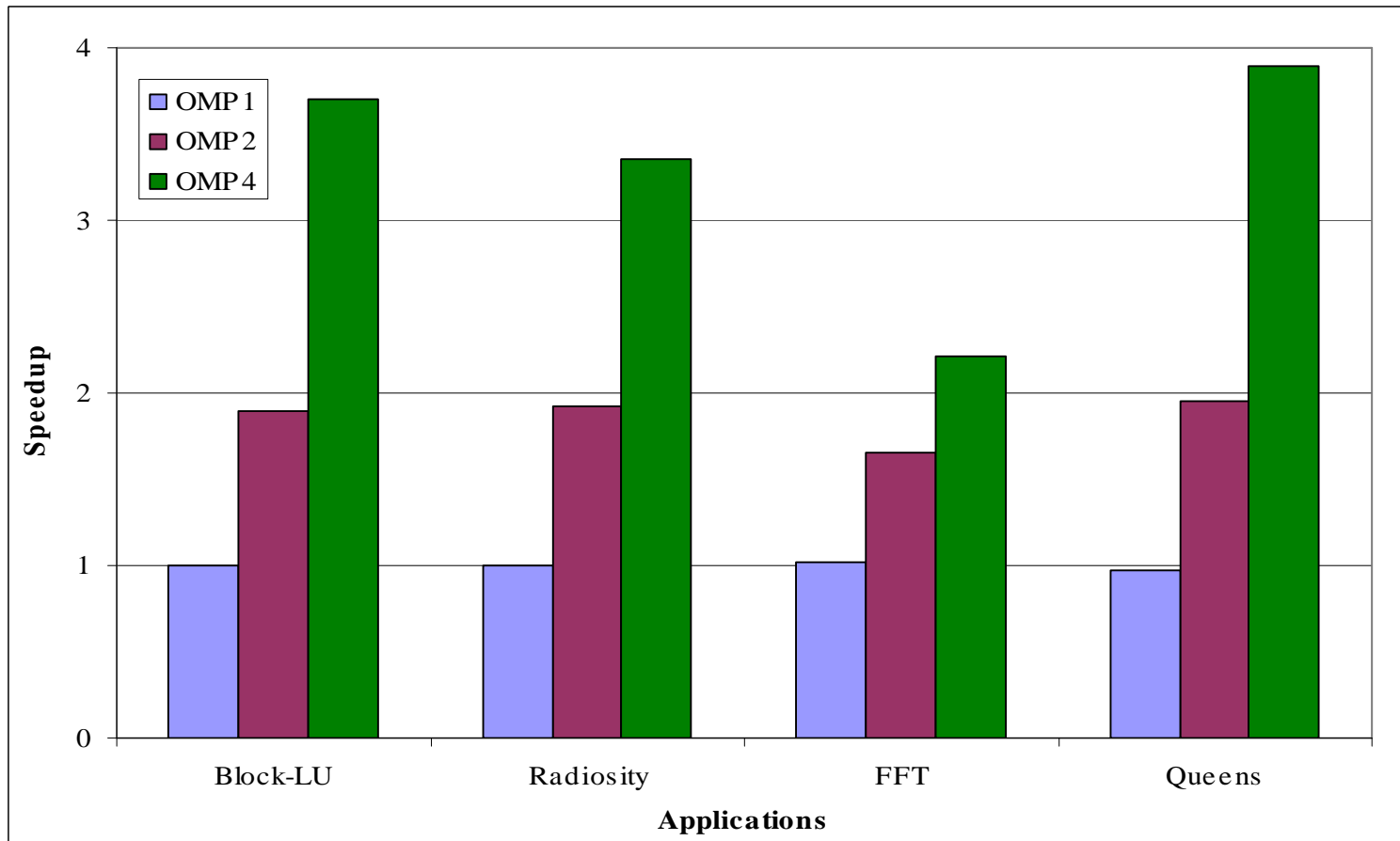
NPB 2.3 (Omni/NthLib)



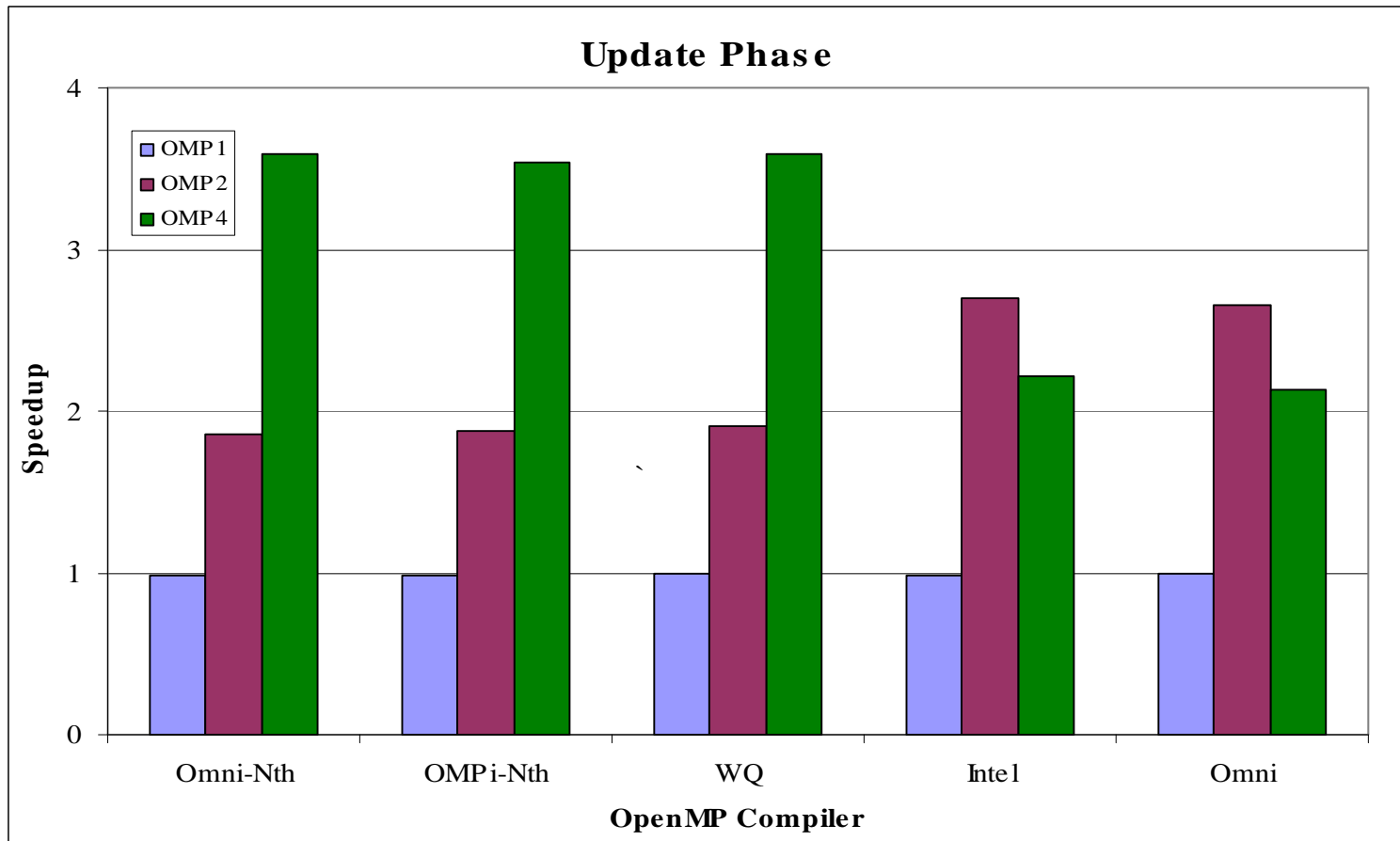
NPB 2.3 (OMPI/NthLib)



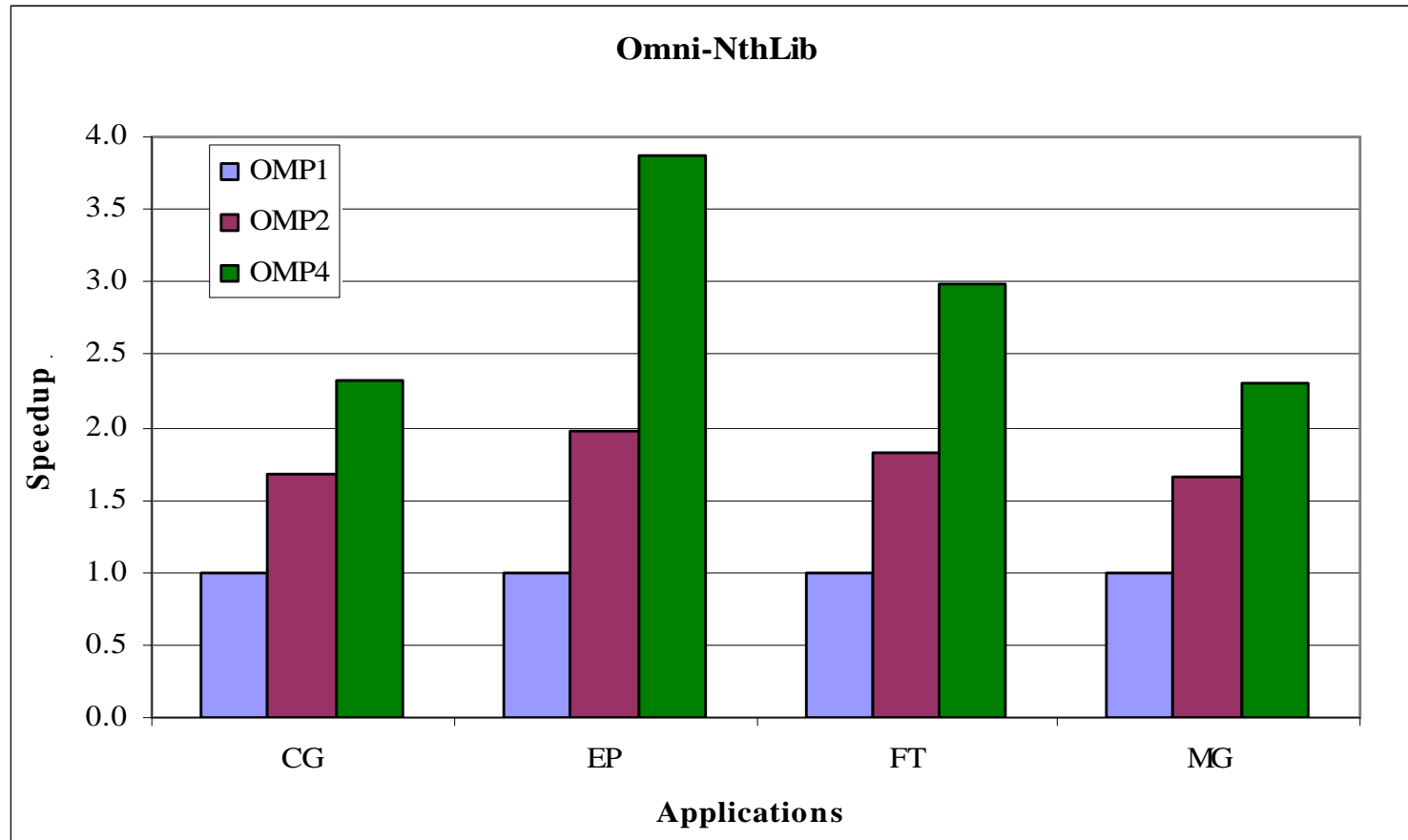
Workqueuing benchmarks (NthLib)



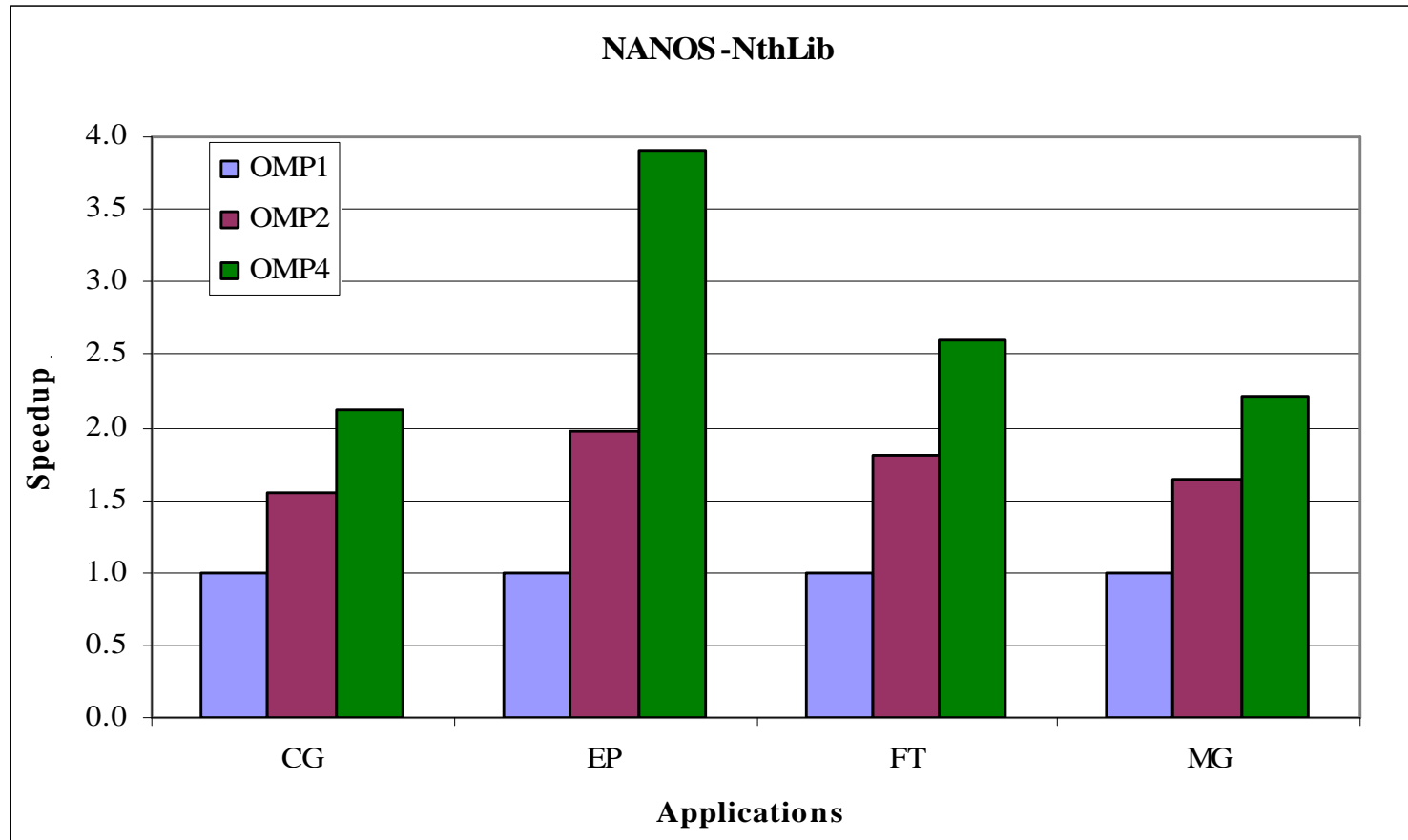
Parallel CURE (Update phase)



NPB 3.0 Fortran (Omni/NthLib and gcc)



NPB 3.0 Fortran (NANOS and g77)



Summary

- Some issues
 - Inefficiency of kernel threads
 - Benchmarks for nested parallelism
- Ongoing work
 - CPU Manager (interaction of kernel and user-level scheduler)
 - OpenMP on SDSM and PCURE
- Acknowledgments
 - Dr. Clay Breshears
 - NANOS project
 - POP project (www.cepba.upc.es/pop)