

# Multicore processors

Trends, problems and solutions

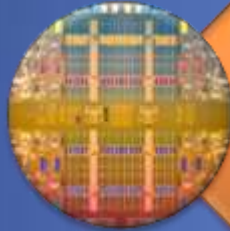
Mats Brorsson SICS & KTH  
matsbror@sics.se, matsbror@kth.se



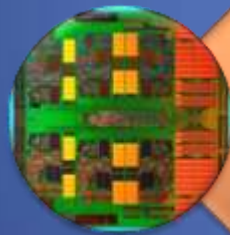
# 2010 state-of-the-art, high-end



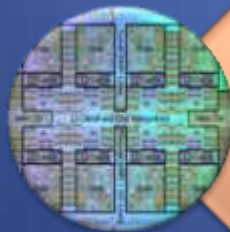
Intel Sandybridge, 6 cores,  
12 threads, 130 W



Intel Xeon 7500, 8 cores, 16  
threads, 130 W



AMD Magny Cours  
12 cores, 12 threads, 105 W



IBM POWER7, 8 cores, 32  
threads, ? W

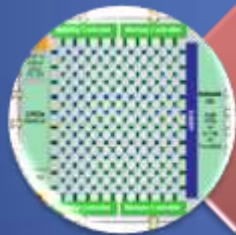
# 2010 state-of-the-art, embedded



Freescale P4080 8 cores,  
30 W



IBM/Sony/Toshiba Cell  
BE, 9 cores, 40 W

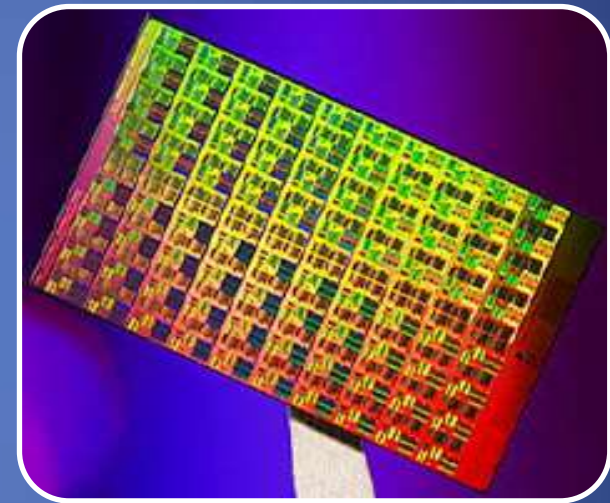
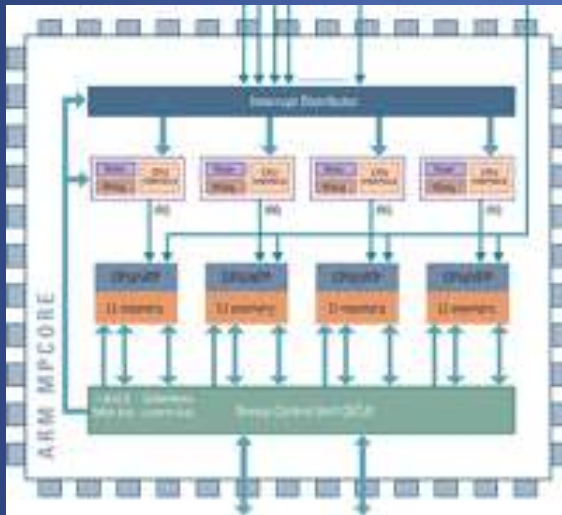


Tiler Tile-GX, 100 cores,  
50 W



Octeon cn68xx, 32 cores,  
?W

# We're going multicore



Software development

==

Development for  
Multicore



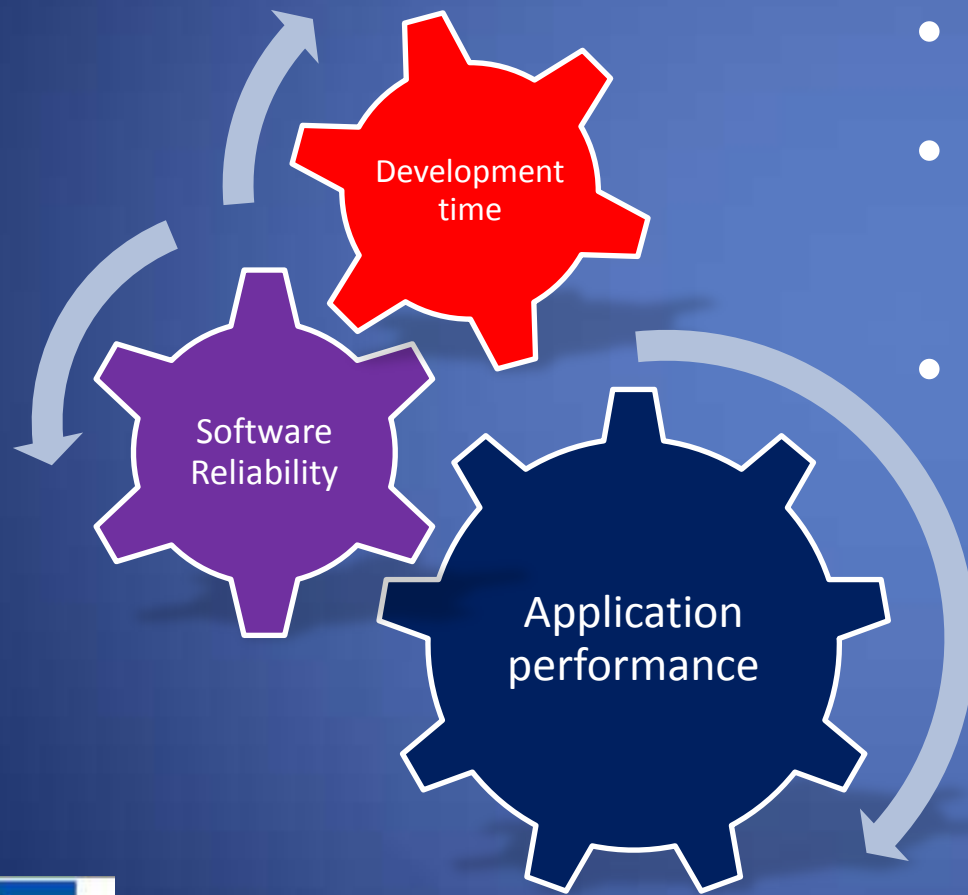
# Some observations...

- Amount of cores is rapidly rising
- I/O-devices, controllers and accelerators are moving onto the same die as processors
- So far, shared address space
  - With some exceptions
- So far, homogeneous architectures
  - With some exceptions

# The Multicore Software Triad

Desirable properties:

- Scalable up and down
- Do not introduce new bugs with parallelism
- Easy to write and maintain



# Programming model design space

- General-purpose languages (GPL)
  - Native threads/processes
  - Data parallelism
  - SPMD
  - Loop-based parallelism
  - Task-based parallelism
- Domain-specific languages (DSL)
  - Erlang/SDL/Matlab/Labview
  - GPU-languages
- Automatic parallelization
- Tool support

# A coding example

## Serial code

```
int fib(int n){
    if (n < 2) return n;
    else {
        int x = fib(n-1);
        int y = fib(n-2);
    }
    return x + y;
}

int main(int argc,
          char *argv[]){
    int n = atoi(argv[1]);
    int result = fib(n);
}
```

- Fibonacci numbers
- Stupid algorithm
- Good didactic example

# Homegrown threading: pthreads

```
int fib(int n){
    if (n < 2) return n;
    else {
        int x = fib(n-1);
        int y = fib(n-2);
    }
}

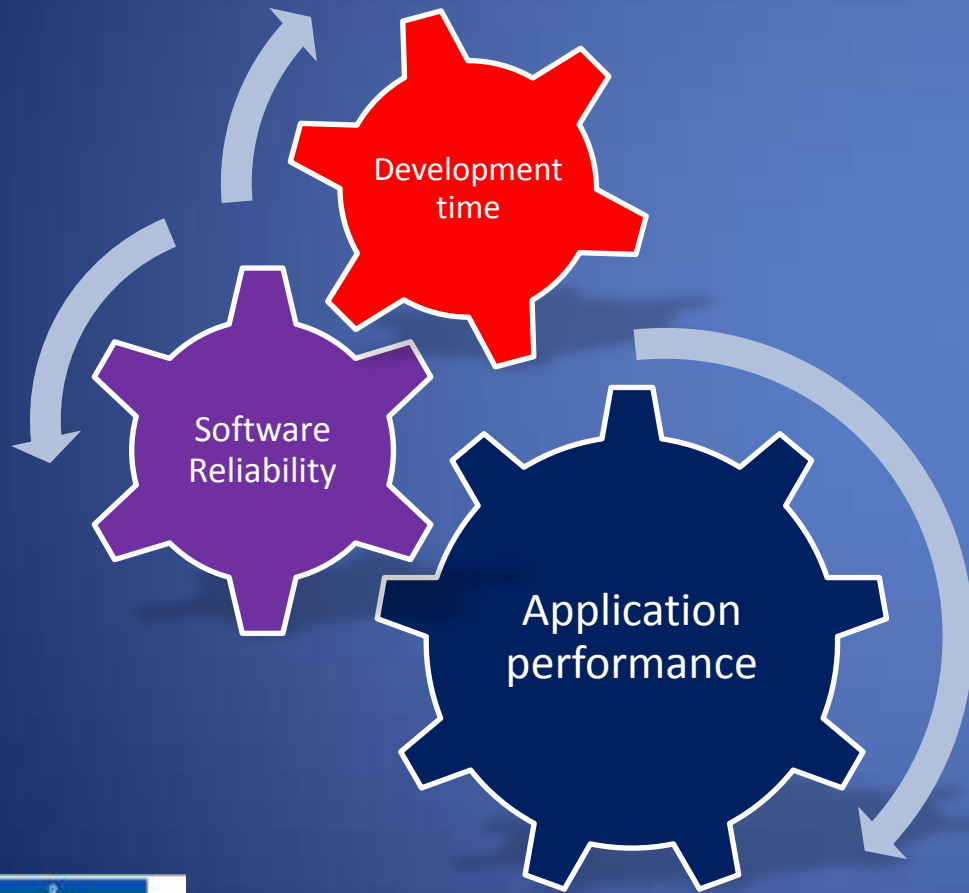
typedef struct {
    int input;
    int output;
} thread_args;

void *thread_func(void *ptr){
    int i=((thread_args *)ptr)->input;
    ((thread_args *)ptr)->output =
        fib(i);
    return NULL;
}
```

```
int main(int argc; char *argv[]){
    pthread_t thread;
    thread_args args;
    int status, results, thread_res;
    int n = atoi(argv[1]);
    if (n < 30) result = fib(n);
    else {
        args.input = n-1;
        status = pthread_create(
            &thread,
            NULL,
            thread_func,
            (void*)&args);

        result = fib(n-2);
        pthread_join(thread, NULL);
        result += args.output;
    }
}
```

# Consequences of pthreads



Desirable properties:

- Scalable up and down
  - Code is fixed to two threads
- Do not introduce new bugs with parallelism
  - Code no longer modular
- Easy to write and maintain
  - 50 year leap backwards!

# Same example using OpenMP

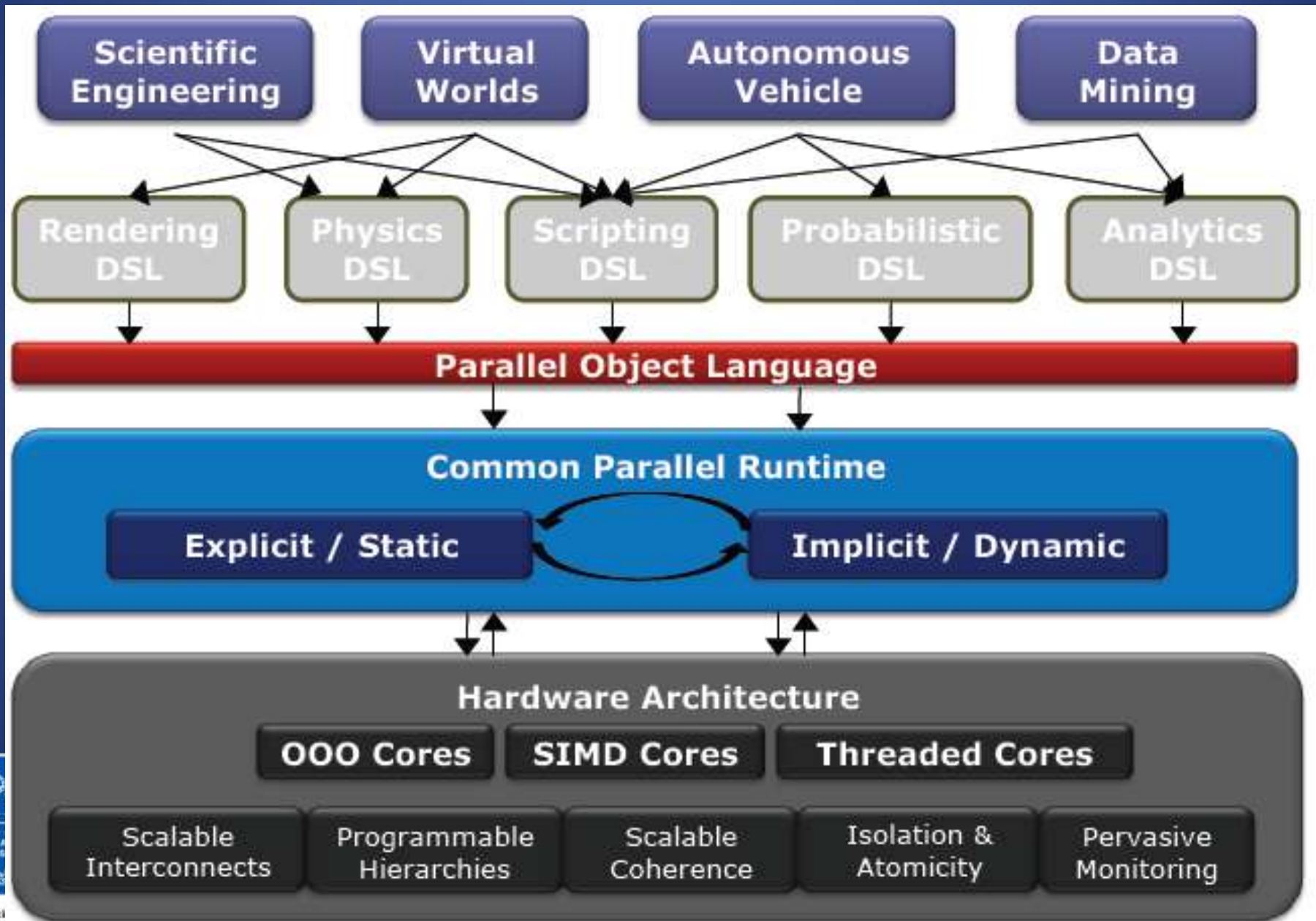
```
int fib(int n){
    int x, y;
    if (n < 2) return n;
    else {
#pragma omp task shared(x)
        x = fib(n-1);
#pragma omp task shared(y)
        y = fib(n-2);
#pragma omp taskwait
    }
    return x + y;
}
```

```
int main(int argc; char *argv[]){
    int n = atoi(argv[1]);
    if (n < 30) result = fib(n);
    else {
#pragma omp parallel
#pragma omp single
        result = fib(n);
    }
}
```

# The Parallel World according to Stanford University

- Automatic parallelization of sequential programs
  - Static Compiler => not general
  - Hardware => too complex
- Hardware + Dynamic Compiler => OK, but won't scale
- Explicit parallel programming
  - Threads, locks
    - Pthreads, OpenMP, MPI
  - Too difficult find parallelism, to debug, maintain and get good performance for the masses
  - Low productivity
  - Limits forward scalability
- Need a new direction

# The PPL Vision: Domain specific languages



# Kista Multicore Center

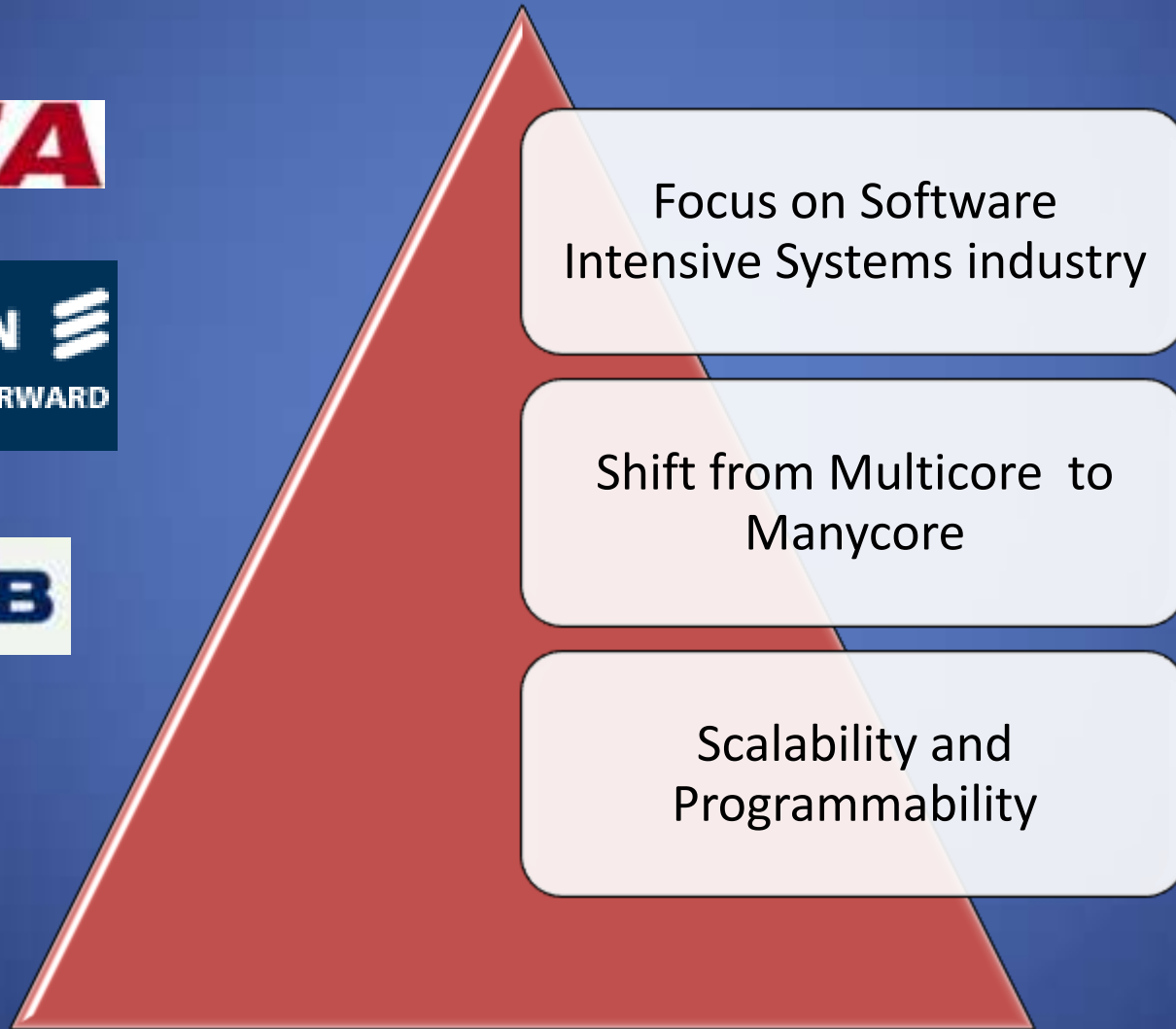
The Multi/Manycore technology is the semiconductor solution to keep up Moore's law

Significant effort necessary to shift to parallel computing and the manycore technology opens up new problems

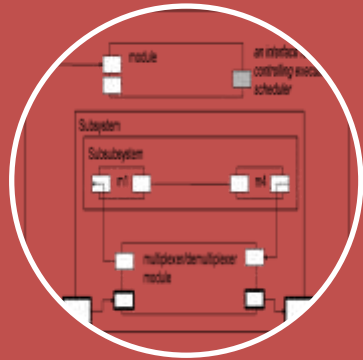
Multicore Center in Kista aims to grow to become a national center for research on manycore technology issues with an *industrial focus*

- Complements other centers, such as UPMARC, that have a more *academic road-map*

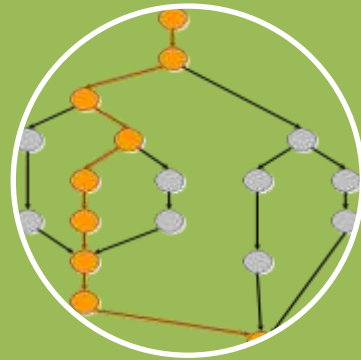
# The IMCORE Project



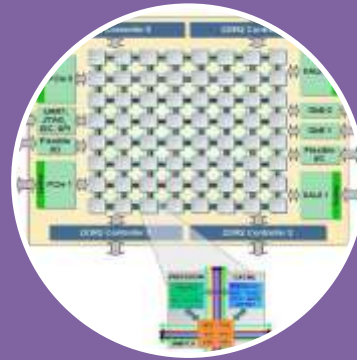
# The four parts of IMCORE



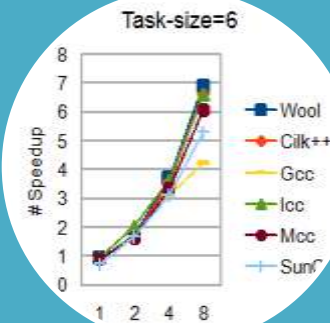
Characterization and definition of application patterns



Execution models of industrial applications on manycore platforms



Application-driven characterization of manycore architectures



Tools and methods for Safe task-based parallel programming



# People

- Mats Brorsson, senior researcher, professor
- Karl-Filip Faxén, senior researcher
- Konstantin Popov, senior researcher
- Vladimir Vlassov, senior researcher, associate professor
- Mladen Nikitovic, PhD student
- Ananya Muddukrishna, PhD student
- Artur Podobas, PhD student
- Farokh Khajuee, MSc student
- Georgious Varisteas, MSc student
- Jianrong Zhang, MSc student
- Niklas Wahlén, BSc student
- Mikael Östberg, BSc student

# Acknowledgements

