PATC Parallel Programming Workshop: Exploring parallelization strategies with Tareador

Eduard Ayguadé, Rosa M. Badia and Vladimir Subotic

Barcelona Supercomputing Center (BSC-CNS)
Universitat Politecnica de Catalunya (UPC-BarcelonaTECH)

Barcelona. October 15, 2013

Part I

Tareador environment

Outline

Motivation

API and example of use

Usage

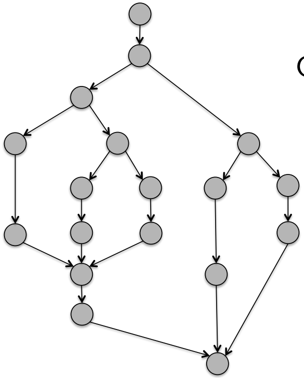
Task decomposition

What is task decomposition?

- From a **sequential** specification of the program ...
- ... find a decomposition of the problem in **tasks** (i.e. identify pieces of work that can execute concurrently) ...
- ... ensuring that the same result is produced (i.e. identify **dependencies** that impose ordering and data sharing constraints).

These tasks and constraints can be later mapped to the elements offered by parallel programming languages

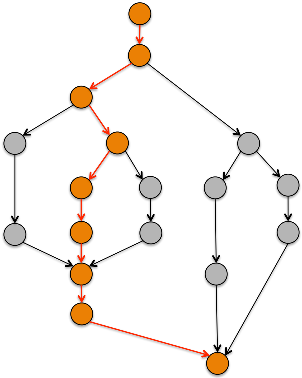
Understanding the potential of a task decomposition



Computation **task graph** abstraction

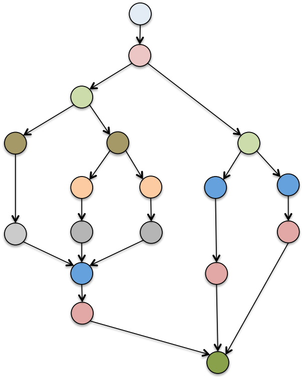
- Directed Acyclic Graph
- Node = dynamic instance of an annotated task (tracking of task entry/exit)
- Edge = dependence between tasks (tracking of dynamic allocations and memory accesses)

Understanding the potential of a task decomposition



- $T_1 = \sum_{i=1}^{nodes}(work\_node_i)$
- $T_\infty = \sum_{i \in criticalpath}(work\_node_i)$, assuming sufficient resources
- Parallelism $= T_1/T_\infty$

Understanding the potential of a task decomposition

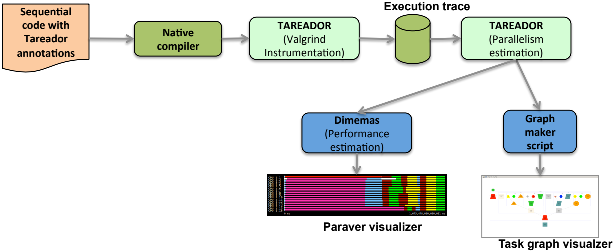


- T_p = execution time on P processors (depends on the schedule of the graph nodes on the processors)
- Speedup on P processors: $S_p = T_1/T_p$

Understanding the potential of a task decomposition

Tareador environment

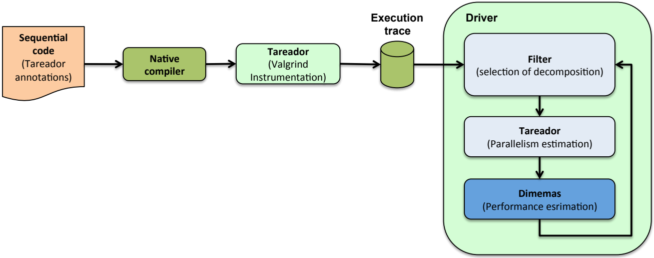
- API to annotate sequential program with potential tasks
- Binary instrumentation using a new **Valgrind** module
- Visualization of task graph (granularities and dependences)
- Simulation with **Dimemas** and visualization with **Paraver**



Exploration of potential task decompositions (next release)

Tareador explorer

- Nesting of potential tasks in sequential program
- Binary instrumentation and execution (only once)
- Hierarchical exploration of task decompositions
- Estimation of parallelism and execution simulation (**Dimemas**)



Outline

Tareador API

- Specification of tareador region

```
tareador_ON();
     ...
tareador_OFF();
```

- Specification of task boundaries

```
tareador_start_task("name of task");
     /* Code region / task */
tareador_end_task();
```

Nesting of tasks (e.g. due to recursion) is possible

- Filtering objects

```
tareador_disable_object(address of object);
     /* Code region */
tareador_enable_object(address of object);
```

Example: dot product

Sequential source code for iterative dot product:

```
void dot_product (long N,
    double A[N], double B[N], double *acc){
    double prod;

    *acc=0.0;
    for (int i=0; i<N; i++) {

        prod = A[i]*B[i];
        *acc+= prod;

    }
}
```

```
int main() {


    for (int i=0; i< N; i++) A[i]=i;


    for (int i=0; i< N; i++) B[i]=2*i;


    dot_product (N, A, B, &result);


}
```

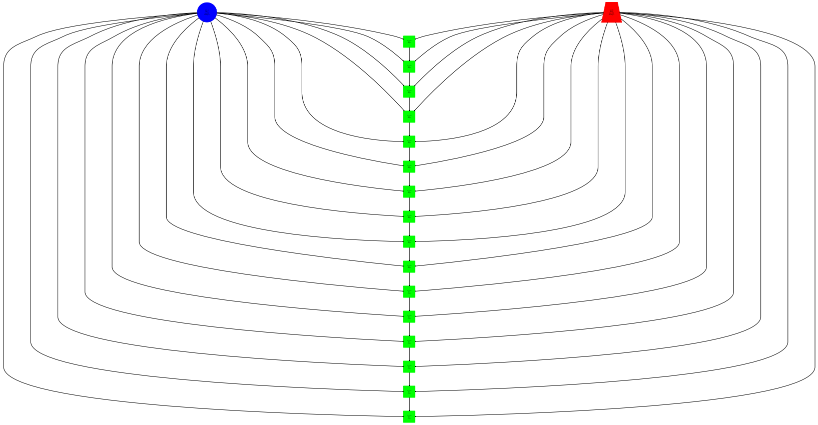
Example: dot product

Instrumented source code for iterative dot product:

```
void dot_product (long N,
    double A[N], double B[N], double *acc){
    double prod;

    *acc=0.0;
    for (int i=0; i<N; i++) {
        tareador_start_task("inner_product");
        prod = A[i]*B[i];
        *acc+= prod;
        tareador_end_task();
    }
}
```

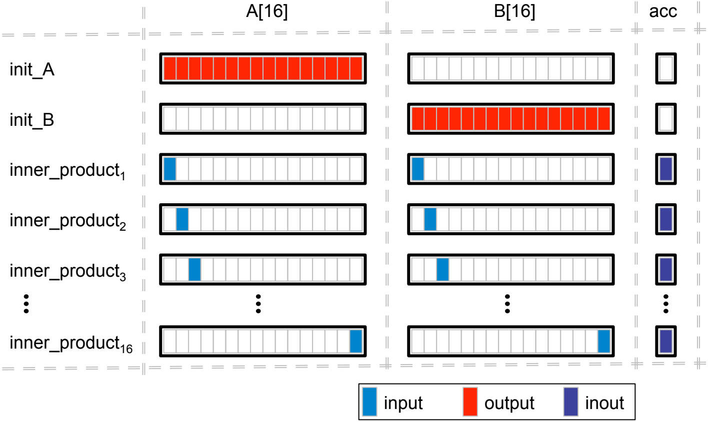
```
int main() {
    tareador_ON ();

    tareador_start_task("init_A");
    for (int i=0; i< N; i++) A[i]=i;
    tareador_end_task();

    tareador_start_task("init_B");
    for (int i=0; i< N; i++) B[i]=2*i;
    tareador_end_task();

    dot_product (N, A, B, &result);

    tareador_OFF ();

}
```

Dot product: task graph, N=16



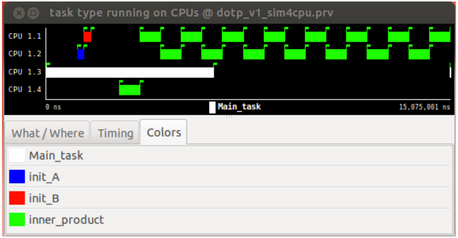
Dot product: task graph, N=16 (next release)

- Data access

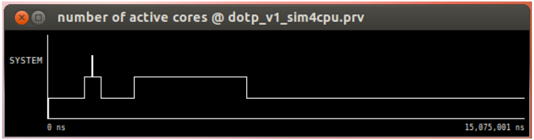


Dot product: Dimemas simulation, N=16, 4 CPU

- CPU view load configuration



- Maximum concurrency view configuration



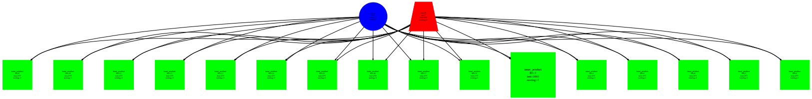
Example: dot product (cont.)

Instrumented source code (filtering object *acc*) for iterative dot product:

```
void dot_product (long N,
    double A[N], double B[N], double *acc){
    double prod;

    *acc=0.0;
    for (int i=0; i<N; i++) {
        tareador_start_task("inner_product");
        prod = A[i]*B[i];
        tareador_disable_object(acc);
        *acc+= prod;
        tareador_enable_object(acc);
        tareador_end_task();
    }
}
```

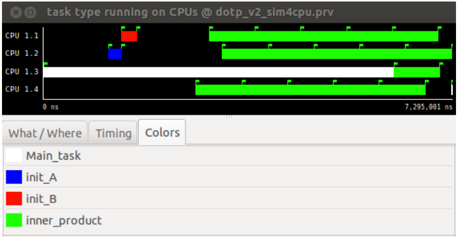
```
int main() {
    tareador_ON ();

    tareador_start_task("init_A");
    for (int i=0; i< N; i++) A[i]=i;
    tareador_end_task();

    tareador_start_task("init_B");
    for (int i=0; i< N; i++) B[i]=2*i;
    tareador_end_task();

    dot_product (N, A, B, &result);

    tareador_OFF ();
}
```

Dot product: task graph filtering *acc*, N=16

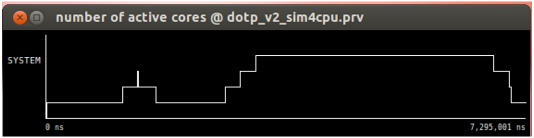


Dot product: Dimemas simulation filtering *acc*, N=16

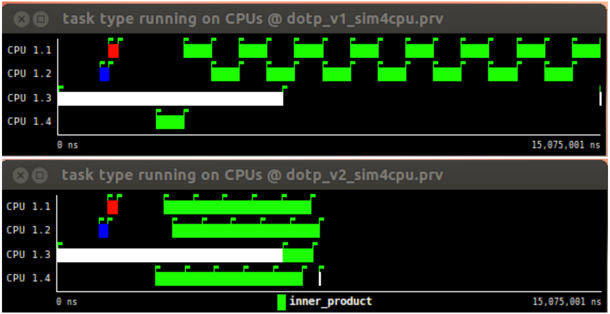
- CPU view load configuration



- Maximum concurrency view configuration



Dot product: Dimemas simulation comparison, N=16



- Comparison at same time scale

Outline

Two usage modes

- **Web portal**[1], execution on the Grid
  - Useful for interactive hands-on and demo sessions
  - Reduced functionality (selection, visualizers, ...)
  - Timeouts in each step of the process
  - Some limitations (single file, no input files, ...)
- Linux command line[2]
  - Installation of tarball or OVF image
  - Examples and Makefile for compilation
  - Scripts for **Tareador** execution and **Dimemas** simulation
  - Configuration files for **Paraver** trace analyzer

[1] `http://bscgrid06.bsc.es/~tareador`

[2] Download from `pm.bsc.es/SC13_HPCEducators`

Under construction ... next features

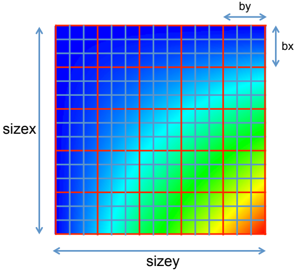
- Improve web portal to remove current limitations
- Information about data dependences and accesses
- Automatic exploration of task decomposition strategies
  - Additional constraints introduced by the programming model (e.g. OpenMP, OmpSs, ...)
- Automatic generation of parallel code

Part II

Hands-on with Heat Equation

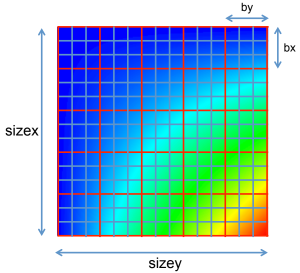
Code 1: Heat equation

```
iter = 0;
while(1) {
    switch( param.algorithm ) {
        case 0: // JACOBI
            residual = relax_jacobi(param.u, param.uhelp, np, np);
            // Copy uhelp into u
            for (i=0; i<np; i++)
                for (j=0; j<np; j++)
                   param.u[ i*np+j ] = param.uhelp[ i*np+j ];
            break;
        case 1: // GAUSS
            residual = relax_gauss(param.u, np, np);
            break;
        case 2: // RED-BLACK
            residual = relax_redblack(param.u, np, np);
            break;
    }
    iter++;
    // solution good enough ?
    if (residual < 0.00005) break;

    // max. iteration reached ? (no limit with maxiter=0)
    if (param.maxiter>0 && iter>=param.maxiter) break;
}
```



Code 1: Heat equation (Jacobi solver)

```
double relax_jacobi (double *u, double *utmp,
                     unsigned sizex, unsigned sizey) {

double diff, sum=0.0;
int nbx, bx, nby, by;

nbx = NB; bx = sizex/nbx;
nby = NB; by = sizey/nby;

for (int ii=0; ii<nbx; ii++)
   for (int jj=0; jj<nby; jj++)
      for (int i=1+ii*bx; i<=min((ii+1)*bx, sizex-2); i++)
         for (int j=1+jj*by; j<=min((jj+1)*by, sizey-2); j++) {
            utmp[i*sizey+j]= 0.25 * (u[ i*sizey     + (j-1) ]+
                                     u[ i*sizey     + (j+1) ]+
                                     u[ (i-1)*sizey + j     ]+
                                     u[ (i+1)*sizey + j     ]);
            diff = utmp[i*sizey+j] - u[i*sizey + j];
            sum += diff * diff;
         }
return sum;
}
```



Code 1: Heat equation (hands-on outline)

1. Try the initial task definition (Jacobi solver, one task per computation of block)
2. Which dependence is causing the serialization of all tasks?
3. Filter the object that causes the dependence (`tareador_disable_object` and `tareador_enable_object`)
4. Dimemas simulation with different number of processors
5. What else could be parallelized (or sequentially optimized)?
6. Repeat process with Red–Black solver
7. Repeat process with Gauss-Seidel solver

PATC Parallel Programming Workshop: Exploring parallelization strategies with Tareador

Eduard Ayguadé, Rosa M. Badia and Vladimir Subotic

Barcelona Supercomputing Center (BSC-CNS)
Universitat Politecnica de Catalunya (UPC-BarcelonaTECH)

Barcelona. October 15, 2013