1 A data-flow runtime

1. Download the archive `dfruntime.tar.gz` from the course's web site. It supports the TStar data-driven execution model described in the course.

2. Compile the library and run the test example on a GNU/Linux distribution with gcc 4.4 or later; there are a few warnings related with GOMP, gcc's OpenMP implementation.

3. Understand as much of the code as possible.

   The part of the code with the `GOMP`-prefixed function calls is a rather convoluted and non-portable way to wrap all the `main()` function of a C program into an OpenMP `parallel` region.

   The pragma syntax `#pragma omp task` marks the next statement/block for concurrent execution as an OpenMP task. Using this syntax, the implementation of `tdecrease()` immediately spawns the dependent task when its synchronization counter reaches 0. The underlying OpenMP runtime deals with the scheduling of independent tasks, whose semantics is very similar to Cilk threads.

4. Starting from the following sequential implementation of the merge-sort algorithm, take inspiration from the test example of the data-flow runtime to write a parallel data-flow version of the merge-sort algorithm.

```
extern int garray[200000];

extern int merge(int, int, int, int, int);

int m_sort(int left, int right)
{
  int mid;
  int a, b, d, f;

  if (right <= left)
      return 0;

  mid = (right + left) / 2;
  a = m_sort(left, mid);
  b = m_sort(mid+1, right);

  f = merge(a, b, left, mid+1, right);

  return f;
}

int main ()
{
  int i = 1;
  int size = sizeof (garray)/sizeof(garray[0]);

  i = m_sort (i-1, size-1);
}
```

2 Scheduling

1. Download the manual and source code for Cilk 5.4.6 from `http://supertech.csail.mit.edu/cilk`

2. Write a parallel Cilk implementation of the matrix product realizing $\mathcal{O}(n^3)$ operations but with a critical path of $\mathcal{O}(\log n)$.

3. Look at more examples from the `examples/` directory of Cilk 5.4.6.

4. Download the paper on work-stealing by Blumofe and Leiserson from the course's web site.

5. Read Section 3 and understand the proofs.

6. Sketch an example of a Cilk program whose sequential execution uses a bounded amount of memory, but where an ad-hoc scheduling strategy uses an amount of memory proportional to the number of instances of a given task.

7. Read the example, discussion and different design choices about cactus stacks in Cilk and Intel's TBB:

   `http://software.intel.com/en-us/articles/the-thorny-problem-of-the-cactus-stack`

8. Vector Fabrics' vfTasks is a simple task library supporting inter-task dependences, with a control flow semantics (as opposed to the data-driven semantics of the data-flow runtime above):

   `http://sourceforge.net/projects/vftasks`