Parallel PARI

B. Allombert

IMB
CNRS/Université Bordeaux 1

17/01/2013
Introduction

An example problem
  The simplest parallel solution

The experimental GIT branch bill-mt
  Examples

The libpari interface
  Example : Code of pareval

Low-level PARI POSIX thread interface
We add support for two common multi-threading technologies:

- POSIX thread: run on a single machine, lightweight, fragile.
- Message passing interface (MPI): run on as many machines as you want, robust, heavyweight.
An example problem

We want to compute the value of a function for all integers less than 1000. Each call takes 1 hour.

```plaintext
for(i=1, 1000, print(i, " : ", fun(i)))
```

This will take 1000 hours.
Lignes directrices

Introduction

An example problem
  The simplest parallel solution

The experimental GIT branch bill-mt
Examples

The libpari interface
  Example : Code of pareval

Low-level PARI POSIX thread interface
The simplest parallel solution

Now assume we have a MPI cluster with 100 cores at our disposal. We rewrite the program as follow:

```plaintext
N=eval(getenv("OMPI_COMM_WORLD_RANK"));
for(i=10*N+1,10*N+10,
    write(Str("fun",N,),i,"":"",fun(i)))
```

We launch it using OpenMPI mpirun command:

```
mpirun -np 100 gp fun.gp
```

Your computation will be finished in 10 hours, the results split in the files `fun0` to `fun99`. 
The experimental GIT branch bill-mt

- New Configure flag: -mt=single, -mt=pthread, or -mt=mpi
- New GP default nbthreads
- New GP functions pareval, parapply, parvector, parsum
Parallel functions

- parvector: parallel version of vector
- parapply: parallel version of apply
- parsum: parallel version of sum
- pareval: evaluate a vector of closure in parallel
Lignes directrices

Introduction

An example problem
The simplest parallel solution

The experimental GIT branch bill-mt
Examples

The libpari interface
Example : Code of pareval

Low-level PARI POSIX thread interface
Examples

res=parvector(1000,i,fun(i));
s=parsum(i=1,1,1000,fun(i));
a=parapply(fun,[1..1000]);
c=pareval(vector(1000,i,()→fun(i)));

This assumes the function fun() does not have side-effect.
The libpari interface

- `handle = mt_queue_start(worker)` Return a handle for parallel evaluation of worker.

- `mt_queue_submit(handle, workid, work)` Submit work to be evaluated by worker, assigning the id workid.

- `result = mt_queue_get(handle, &workid, &pending)` Return the evaluation by worker of some of the previously submitted works. Set pending to the number of remaining pending works, and workid to the id of the job.

- `mt_queue_end(handle)` Free the resource allocated by handle and end the parallel execution.

Call to `mt_queue_submit` and `mt_queue_get` must be alternated.
Lignes directrices

Introduction

An example problem
  The simplest parallel solution

The experimental GIT branch bill-mt
Examples

The libpari interface
  Example : Code of pareval

Low-level PARI POSIX thread interface
Example : Code of pareval

GEN pareval_worker(GEN C)
{
    return closure_callgenall(C, 0);
}
GEN pareval(GEN C)
{
    pari_sp av = avma;
    long l = lg(C), i, pending = 0, workid;
    GEN worker = snm_closure(is_entry("_pareval_worker"),
        void *mt = mt_queue_start(worker);
    GEN V = cgetg(l, t_VEC), done;
    for (i=1; i<l || pending; i++)
    {
        mt_queue_submit(mt, i, i<l? mkvec(gel(C,i)): NULL);
        done = mt_queue_get(mt,&workid, &pending);
        if (done) gel(V,workid) = done;
    }
    mt_queue_end(mt);
    return gerepilecopy(av, V);
}
Low-level PARI POSIX thread interface

You need to use `Configure --enable-tls`. See Appendix D of the manual, and the file `example/thread.c`

Parent thread:

- `pari_thread_alloc()` Allocate a PARI stack for a thread.
- `pari_thread_free()`

Child thread:

- `pari_thread_start()` Initialize threads using the specified stack.
- `pari_thread_close()`