Making High Performance Computers Highly Productive

Bill Carlson
IDA Center for Computing Sciences
What's the state of high end computing?

Certainly not the best of times

What does “productivity” really mean?

Measurements?

A few ideas on a way forward
What's right

- We are getting many, many more cycles
  - With respect to ops, Moore rules
- Memory bandwidth is starting to improve
  - Cray X1, Alpha EV7, Opteron, Others
- We are getting some new applications
  - But not enough
What's Wrong

- Still no stable market
- Programming time is increasing
  - No real solution for parallel programming
- Decreasing Numbers:
  - Users
  - Programmers
  - Institutions who care
The Impact Zone

Capability Increase

Impact

Delay
Observations and Trends

- Delay to impact zone is too long
  - And growing
- Length of impact zone is short
  - Excitement wanes quickly
- Depth of impact zone is shallow
Productivity Today

Ideal
Application
Programming
HPC Programming 1980
HPC Programming Today

Productivity
Skill
0?
Productivity

- Make the impact zone sooner, longer, deeper
- Economics 101: Output per unit of input
  - GDP per unit labor-hour
  - Output is not ops, but impact
  - Input is not just $, but human capital
- Real output of HPC is Understanding
- Need to find the GDP for our field!
Measuring is Hard

- Programming time:
  - Usually measured in SLOC/time
  - This is good for project managers
  - Very BAD for trying to tune productivity
    - presumes constant level of abstraction
  - OK, measuring run time is “easy”
- Interpretation time?
- All DEPENDENT variables!
One Measurement Idea

Computational Mass/Action (Bob Numrich)

Physics isomorphism:

- work, distance, time -> ops, bits, hz
- computational mass is derived: ops/(bit-hz)^2
- Newton’s Laws (and much of physics) can be used!

Programming and Interpretation Work

- Code/Data represented has some mass
- Force needed to move from “bad” “location” to “good”

Needs work, but hope for unification
A way forward

- Scientific Productivity Stewardship?
- Use science and engineering to guide us
  - Research: New ideas!
  - Vendors: Prototypes VERY Important!
  - Government: Getting it together
    - DARPA HPCS
    - IHEC, NAS study, HEC/RTF
Some Initial Ideas

- Abstraction
  - Type less, reuse more
- Persistence
  - Data lives a long time, why not use it?
- Diversity of Expression
  - No one right way to say everything
Abstraction

Not just about “object oriented” languages

- Abstraction about parallelism (UPC)
- Abstraction about functionality (MatLab)
- Abstraction about data (Transactions)

Abstraction can’t mean low performance!

- Runtime a big component of productivity

Re-Use
NOT only software!

- Architecture
  - More than “shared memory” or “message passing”
- Hardware Assist
  - Better latency/bandwidth always needed
  - Content addressable memory?
Persistence

- Some early experiments are encouraging
  - “Data Mining” and “Object Stores”
- Databases integrated with simulations
- Persistence as a parallel model
  - Transactions help tolerate both latency and faults (both hardware and software)
Integration

- Procedural programming
  - Tell the computer the order to execute
  - Much research on integration here (babel)
- Declarative programming
  - Tell the computer what you want (SQL)
- Never work well together
An Example - “autosql”

- Establish correlation between database tables and data structures in memory.
- “Queries” and “Updates” in database are now automatic (Abstraction)
- Program can “live” forever (Presistence)
  - Automatic checkpoint and restore
- Many instances leads to parallel program
  - Database could be integrated with program
autosql example

```c
def struct mine t {int a, time_t b, double d};
struct auto_sql tbl { "select a,b from c",
    {"a", A_INT, A_OFF (t,a)},
    {"b", A_DATE, A_OFF (t,b)}
};

as = as_open (postgres:test);
dp = as_select (as, &tbl, "where d > %g", 1.4);
if (dp) {
    struct mine *t = dp->data;
    for (i=0; i<dp->n; i++)
    {
        printf ("a:%d b:%s\n", t->a, ctime(t->b));
        t++;
    }
}
```
Summary

High Performance computing is in trouble

Not because of performance growth

Simply not “productive” enough

There is a way forward

Abstraction, Persistence, Integration

I, for one, will start this journey

Please join me!

Summary

Productivity
Skill
Ideal
Application
Programming
HPC Programming 1980
HPC Programming Today