Computing in the Mist: Writing Applications for Unknown Machines

Thilo Kielmann
VU University, Amsterdam
kielmann@cs.vu.nl
THE MIST
NOVEMBER 21
- How many cores does your computer have?
- Where is your data?
- What has happened to job number 2342?
ITRS Roadmap 2005 and 2007

Clock Rate (GHz)

2005 Roadmap

2007 Roadmap

Intel single core

Intel multicore

[D. Patterson, USENIX 2008 keynote]
A Fundamental Technology Change

- CPU's will get faster only marginally:
  - limits of integration density
  - energy consumption (proportional to clock rate)
- If you want a faster computer, you need to use multiple CPU's:
  - In the past, the clock rate has doubled every 18 months
  - In the future, the number of cores will double every 18-24 months
- All programs must be parallel to use this new hardware!
Intel's 80-Core Teraflop Chip Prototype
Intel's Tera Scale Vision
IBM's Cell: Heterogeneous Ensemble
Cell Performance Compared

- FP (SP)
- FP (DP)
- Int (16 bit)
- Int (32 bit)

Freescale MPC8641D 1.5 GHz
AMD Athlon™ 64 X2 2.4 GHz
Intel Pentium D® 3.2 GHz
PowerPC® 970MP 2.5 GHz
Cell Broadband Engine™ 3.2 GHz

[Dr. Michael Perrone, IBM]
Cell Programming: Master/Worker

- PPE executes main program
- SPE's execute sub tasks and return results
- all communication between PPE and SPE's must be programmed explicitly
Example: Nvidia's GeForce8800

- 8 x 16 blocks of stream processors
- separate thread schedulers
- crossbar-like access to graphics memory
GPU vs. CPU

- **Nvidia GeForce 8800**
  - clock speed 1.35 GHz
  - 681 million transistors

- **Intel Pentium 4**
  - clock speed 2.4 GHz
  - 55 million transistors

[Schellmann et.al., Euro-PAR 2008]:

2 Nvidia GeForce 8800GTX as fast as 16 Intel Xeon 3.2GHz
General Purpose GPU Programming

Nvidia's CUDA

Computing $y \leftarrow ax + y$ with a Serial Loop

```c
void saxpy_serial(int n, float alpha, float *x, float *y)
{
    for(int i = 0; i<n; ++i)
        y[i] = alpha*x[i] + y[i];
}

// Invoke serial SAXPY kernel
saxpy_serial(n, 2.0, x, y);
```

Computing $y \leftarrow ax + y$ in parallel using CUDA

```c
__global__
void saxpy_parallel(int n, float alpha, float *x, float *y)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    if( i<n ) y[i] = alpha*x[i] + y[i];
}

// Invoke parallel SAXPY kernel (256 threads per block)
int nblocks = (n + 255) / 256;
saxpy_parallel<<<nbblocks, 256>>>(n, 2.0, x, y);
```
Clusters: Beyond Single Computers

DAS3, VU Amsterdam:
Programming for Clusters

- Distributed memory, high-speed networks
  - commonly: message passing (MPI), C and Fortran
  - academic: Java remote method invocation (Ibis)

- deployment via shared file system and batch queue scheduling
Clouds: Data Centers + Virtualization

(OK, a bit too simplified...)
Cloud Computing: Platform as a Service (PaaS)

- **Amazon Web Services:**
  - **Elastic Compute Cloud (EC2)**
    - allows to dynamically create/remove virtual machines with user-defined image (OS + application)
    - payment for CPU per hour
  - **Simple storage Service (S3)**
    - provides persistent object storage, write-once objects
    - payment for storage volume and transfer volume

- Highly dynamic service provider for compute and storage capacities
Programming Clouds: Hadoop

- implements the map-reduce paradigm
- allows processing of large data sets
- user defined map and reduce functions
- Hadoop distributed file system (HDFS) built on top of Amazon S3
- Got popular due to fault-tolerant, file-based implementation
Grid Computing
int copy_file (char const* source, char const* target) 
{
globus_url_t source_url;
globus_io_handle_t dest_io_handle;
globus_ftp_client_operationattr_t source_ftp_attr;
globus_result_t result;
globus_gass_transfer_requestattr_t source_gass_attr;
globus_gass_copy_attr_t source_gass_copy_attr;
globus_gass_copy_handle_t gass_copy_handle;
globus_gass_copy_handleattr_t gass_copy_handleattr;
globus_io_attr_t io_attr;

int

if ( globus_url_parse (source_URL, &source_url) != GLOBUS_SUCCESS ) {
    printf ("can not parse source_URL \"%s\"\n", source_URL);
    return (-1);
}

if ( source_url.scheme_type != GLOBUS_URL_SCHEME_GSIFTP && source_url.scheme_type != GLOBUS_URL_SCHEME_FTP && source_url.scheme_type != GLOBUS_URL_SCHEME_HTTP && source_url.scheme_type != GLOBUS_URL_SCHEME_HTTPS ) {
    printf ("can not copy from %s - wrong prot\n", source_URL);
    return (-1);
}

globus_gass_copy_handleattr_init (&gass_copy_handleattr);
globus_gass_copy_attr_init (&source_gass_copy_attr);
globus_ftp_client_handleattr_init (&ftp_handleattr);
globus_io_fileattr_init (&io_attr);
globus_gass_copy_attr_set_io (&source_gass_copy_attr, &io_attr);
globus_gass_copy_handleattr_set_ftp_attr (&gass_copy_handleattr, &ftp_handleattr);
globus_gass_copy_handle_init (&gass_copy_handle, &gass_copy_handleattr);

if ( source_url.scheme_type == GLOBUS_URL_SCHEME_GSIFTP ||
    source_url.scheme_type == GLOBUS_URL_SCHEME_FTP ) {
    globus_ftp_client_operationattr_init (&source_ftp_attr);
globus_gass_copy_attr_set_ftp (&source_gass_copy_attr, &source_ftp_attr);
}
else {
    globus_gass_transfer_requestattr_init (&source_gass_attr, source_url.scheme);
}

output_file = globus_libc_open ((char*) target,
    O_WRONLY | O_TRUNC | O_CREAT,
    S_IRUSR  | S_IWUSR | S_IRGRP |
    S_IWGRP);

if ( output_file == -1 ) {
    printf ("could not open the file "%s"
", target);
    return (-1);
}

/* convert stdout to be a globus_io_handle */
if ( globus_io_file_posix_convert (output_file, 0, &dest_io_handle)
    != GLOBUS_SUCCESS) {
    printf ("Error converting the file handle\n");
    return (-1);
}

result = globus_gass_copy_register_url_to_handle (gass_copy_handle, (char*)source_URL,
    &source_gass_copy_attr, &dest_io_handle, my_callback, NULL);

if ( result != GLOBUS_SUCCESS ) {
    printf ("error: %s\n", globus_object_printable_to_string (globus_error_get (result)));
    return (-1);
}

globus_url_destroy  (&source_url);

return (0);
The Grid Application Toolkit (JavaGAT)

Features:

JavaGAT -- A Kindler Gentler Grid Interface
by Rob van Nieuwpoort and Thilo Kielmann
Vrije Universiteit, Amsterdam

[SC'07]
JavaGAT Example: Copy a File
High-level, uniform

```java
import org.gridlab.gat.*;
import org.gridlab.gat.io.File;

public class Copy {
    public static void main(String[] args)
        throws Exception {
        GATContext context = new GATContext();
        URI source = new URI(args[0]);
        URI dest = new URI(args[1]);
        // Create a GAT File object
        File file = GAT.createFile(context, source);
        file.copy(dest); // The actual file copy.
        GAT.end(); // Shutdown the JavaGAT.
    }
}
```

- Provides the high level abstraction, that application programmers need; will work across different systems
- Shields gory details of lower-level middleware system
The need for a standard programming interface
   – Projects keep reinventing the wheel again, yet again, and again
   – MPI as a useful analogy of community standard
   – OGF as the natural choice; established the SAGA-RG

Community process
   – Design and requirements derived from 23 use cases
   – SAGA Design Team (OGF, Berkeley, VU, LSU, NEC)
Ibis: Grid Programming and Deployment Simplified

Grid Applications
MEG Analysis, Multimedia Content Analysis, Satisfiability Solver, Automatic Grammar Learning, N-Body Simulation, etc.

Programming Models
Satin, MPJ, RMI, GMI

IPL
Communication, Membership, Fault Tolerance, etc.

Deployment and Management
IbisDeploy, Adaptive Satin, Barnes GUI, etc.

JavaGAT
Job Submission, Monitoring, File Transfer, etc.

SmartSockets
Robust Communication

Traditional Communication Libraries

Traditional Grid Middleware

Zorilla
Peer to Peer Grid Middleware
Satin: Divide-and-Conquer for the Grid

Effective paradigm for Grid applications (hierarchical)
Satin: Grid-aware load balancing (work stealing)
Also support for
  Fault tolerance
  Malleability
  Migration
Parallel Sudoku Solver with ADLB

Program:
if (rank = 0)
   ADLB_Put initial board

ADLB_Get board
while success (else done)
   ooh
   find first blank square
   if failure (problem solved!)
      print solution
      ADLB_Set_Done
   else
      for each valid value
         set blank square to value
         ADLB_Put new board
   end while

Work-package = partially completed “board”
More: Master-Worker Parallelism

- "Embarrassingly parallel" problems
- minimal communication
- no dependence on numbers/types of computers

- Popular e.g. by Seti-at-home, BOINC, etc.
- only(?) applicable to very simple problems
More: Parallel Skeletons / Higher-Order Components

- Abstract parallelism and communication from the application logic
- Highly useful approach to implement adaptive (autonomic) applications

(If this all reminds you of what you were doing while you were still young, you are getting my point...)
Challenges of Near-future Platforms

- **Scalability**
  - applications have to run on widely different numbers of CPU's
  - if your program can not use twice the number of CPU's, you won't be able to utilize next year's computer

- **Heterogeneity**
  - applications will have to run on many core and multi core, and special-purpose CPU's (like Cell and GPU's)
  - think of clusters of multi core, clusters of Cell's, clusters of clusters of...
Challenges (2)

- **Performance portability**
  - applications must run *efficiently* on different types of machines (one of the hard problems of parallel computing)
  - I mean, both on Tsubame with GPGPU's *and* on Roadrunner with Cell's...

- **Malleability**
  - applications must be able to run with changing numbers of processors, at run time
    - adapt to changing environments

- **Fault tolerance**
  - simple statistics: with a large number of parts involved, failure probability raises towards 1
Lots of heroic efforts squeezing out performance:
- CUDA
- Cell
- Astron writing Assembler for the Blue Gene/L ...

We are back to the (Transputer) times where codes were written for specific parallel machines
- Not what we want (except for researching machines)

The opposite on clouds and grids: (lazy guys)
- Map/Reduce and Hadoop abstract from machine
- but add fault tolerance and malleability
Approach seen by CoreGRID folks

Application + runtime env.

Middleware

Resources
Grid Application Runtime Stack

- **Grid Application Toolkit (GAT)**
- **SAGA**
- **MPICH-G**
- **Workflow**
- **Satin/Ibis**
- **NetSolve**

"just want to run fast"

"want to handle remote data/machines"

Added value for applications

Grid Application Toolkit (GAT)
Approach seen by Berkeley's PAR Lab

[D. Patterson, USENIX 2008 keynote]
Programming Models for the Mist

- We need a lower layer of efficiency primitives that handle certain platforms the respectively “best” way
  - this is for the CCI's and CCR's
  - users should not see this, only tool writers or compilers

- A higher “coordination” layer has to describe available concurrency, in a declarative manner (?)

- This means, we should reconsider the works from the 80s and 90s and see why they failed and what we could use today
  - map-reduce as the perfect example for an old idea, re-animated
  - There is hope for your Ph.D. work, after all ;-)


Summary / Conclusions

- The future is parallel
- Parallel programming is hard
- This is a big chance for Computer Science to get it right, finally...
- My personal take:
  - The solution will be a combination of declarative parallelism, combined with MUCH systems work on getting the plumbing right
  - We might have to step back from getting the last bit of performance in favour of a more sustainable approach