Porting Nanos++ runtime to Barrelfish

Zeus Gómez Marmolejo
Vicenç Beltran Querol
Eduard Ayguadé Parra

Cambridge Barrelfish Workshop

Cambridge, October 21st, 2011

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Motivation

What we are doing

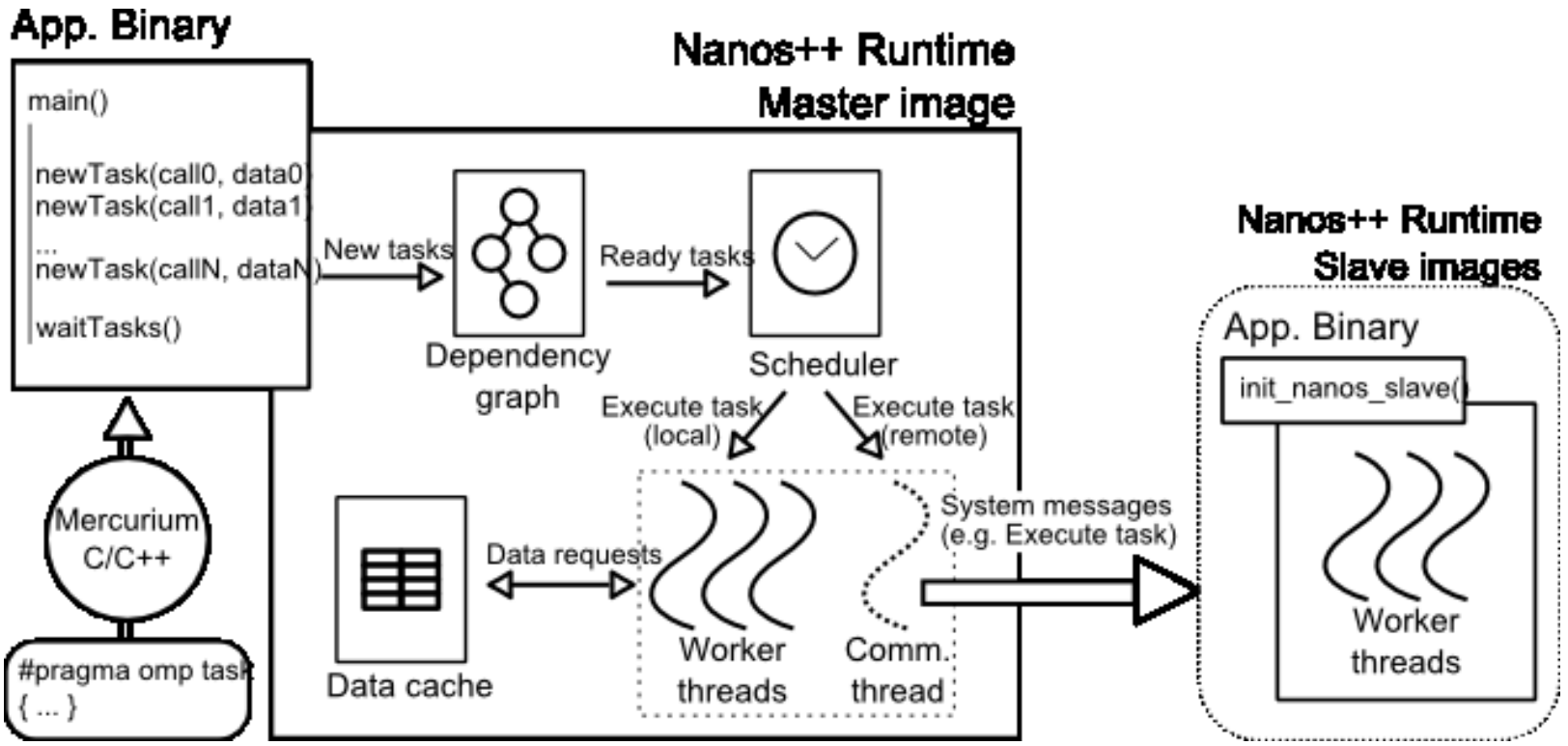
- We are trying to run OpenMP/StarSs programs on Barrelfish, on different architectures
- OmpSs: extends OpenMP with StarSs programming/execution model

```
for (kk=0; kk<NB; kk++) {
   lu0(A[kk][kk]);
   for (jj=kk+1; jj<NB; jj++)
      if (A[kk][jj] != NULL)
         fwd(A[kk][kk], A[kk][jj]);
   for (ii=kk+1; ii<NB; ii++)
      if (A[ii][kk] != NULL) {
         bdiv (A[kk][kk], A[ii][kk]);
         for (jj=kk+1; jj<NB; jj++)
            if (A[kk][jj] != NULL) {
               if (A[ii][jj]==NULL)
                  A[ii][jj]=allocate_clean_block();
               bmod(A[ii][kk], A[kk][jj], A[ii][jj]);
   }}}
```

```
#pragma css task inout(diag[B][B]) highpriority
void lu0(float *diag);
#pragma css task input(diag[B][B]) inout(row[B][B])
void bdiv(float *diag, float *row);
#pragma css task input(row[B][B],col[B][B])
                  inout(inner[B][B])
void bmod(float *row, float *col, float *inner);
#pragma css task input(diag[B][B]) inout(col[B][B])
void fwd(float *diag, float *col);
```

Architecture of Nanos++

- We use Mercurium to convert OmpSs to C++ code with calls to the runtime
- The runtime creates the dependency graph and schedules tasks



Requirements of Nanos++

- Dynamic libraries, loaded “on the fly” (plugins):
  - ✓ Solved using static linking on a different executable section
- TLS (Thread Local Storage)
  - ✓ Now possible in the new version of Barrelfish
- Gasnet library
  - × Barrelfish’ memory leak
- Nanox is written in C++, using an extensive set of C++03 and C++11 features, also with:
  - × Exceptions
  - × Atomic primitives
  - × Complete STL implementation (queues, bitsets,…)

GNU Toolchain and newlib

Current compilation system

Hake controls all the build process

Pros

- Detects everything
  Header files
  Object files
  Libraries, paths,
  Flounder interfaces
- Hakefiles are simple

Cons

- A bit inflexible
  Not compatible with GNU libtool and autoconf
- Existing apps are difficult to port
  The standard libraries and headers are missing

GNU toolchain

Proposal to create a GCC cross compiler and its toolchain. Purpose:

Create static executables that can be directly executed in BF

- **Binutils**. Added new target names, new linker scripts: .text = 0x400000 & .data = 0x600000
- **GCC**. Added:
  - arch spec file, with pre-defines: BARRELFISH
  - default libraries “–lbarrelfish –lc”
  - crt* object files and section placement

Newlib and GNU libstdc++

- **newlib**, as a replacement for libc:
  - Most of the work done here
  - Less code to maintain
  - Complete, thread-safe and totally implemented libc
  - Portable library, to other architectures and OS
- **GNU libstdc++** & **libgcc_eh** (complete C++)
  - Don’t need any change, as they depend only in the libc implementation

New targets

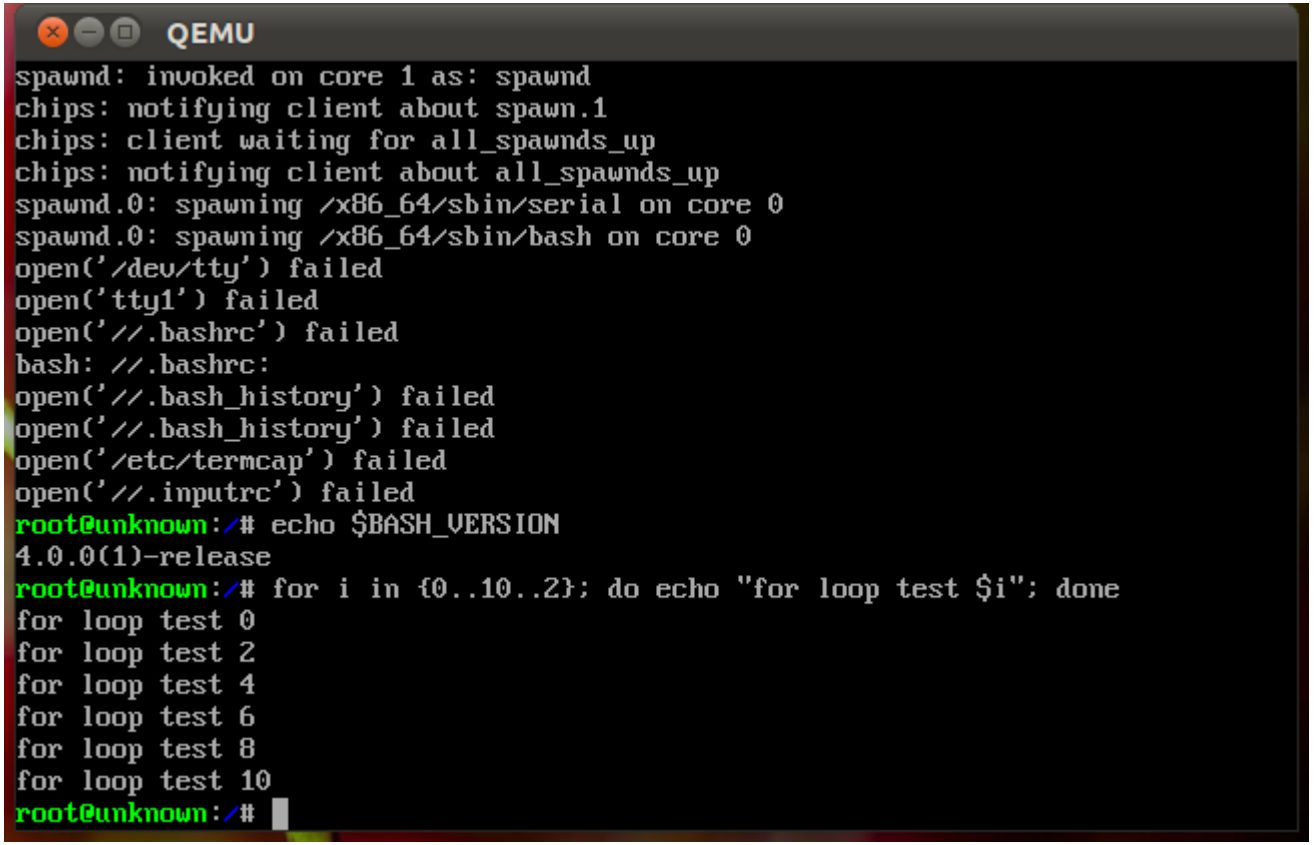
- `x86_64-pc-barrelfish`
- `i586-pc-barrelfish`
- `i586-scc-barrelfish`

To compile a GNU application, it should be as easy as:

```
./configure --host=x86_64-pc-barrelfish
```

Example: GNU bash running in Barrelfish

GNU bash v4.0 running on BF

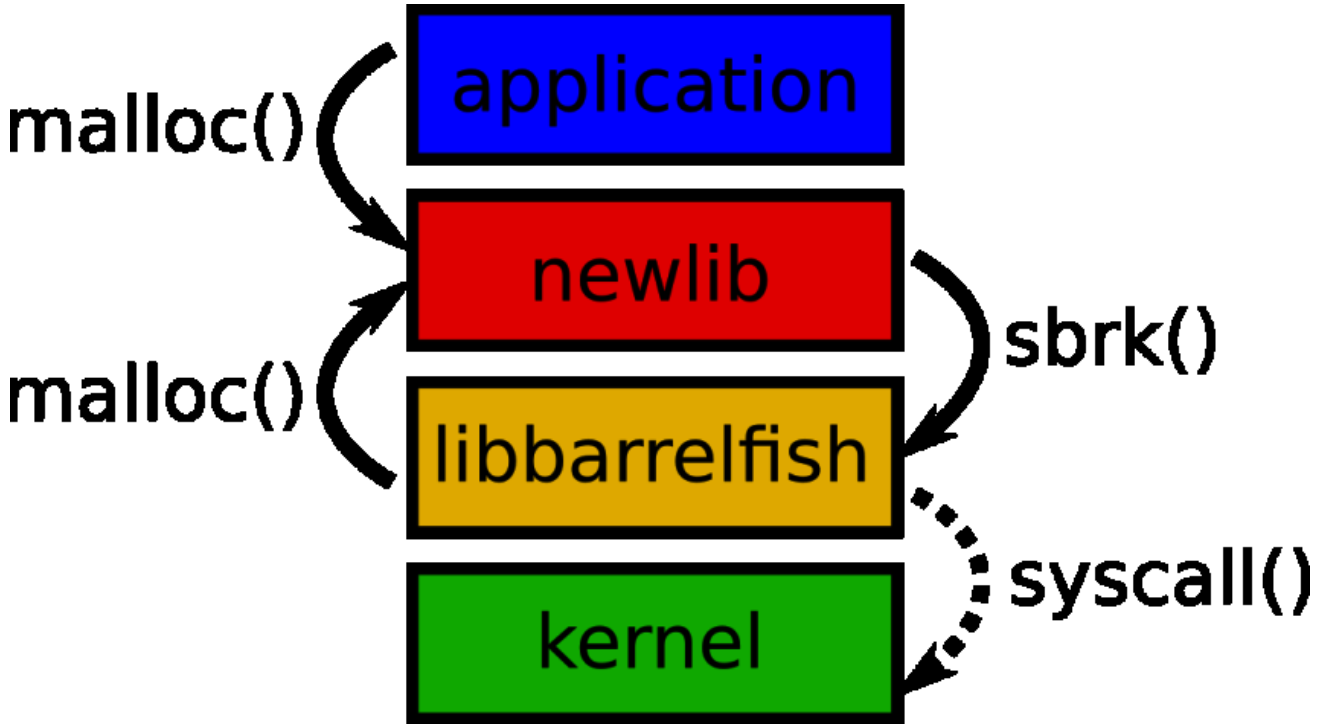


Not everything is so easy

Very basic newlib backend implementation:

- **fork**(), **wait**(), **kill**() and **execve**() not implemented, (so bash is not able to execute any command!) ☺
- Newlib not really prepared for microkernels:
  - One approach is to copy and rename userbase libraries to newlib to avoid naming clashes. **Now broken due to Barrelfish change of version**
  - Rearrange header files and change Barrelfish names in order not to have conflicting types. Store BF build and source directories in GCC config

Example: current malloc() call



Other examples like *fopen()* or *printf()* have similar problems

Other problems happily avoided

```
for(;;) {
  p = malloc(65536);
  free(p);
}
```

The memory leak in the Barrelfish memory allocator can be avoided by:

Using the malloc() libc call present in newlib,

- Relies on sbrk(), the 1st time is allocating a huge space, no additional calls to BF mem alloc needed

All these would be the tools to compile …

Gasnet

What is Gasnet?

- Low-level networking layer for high-performance communication
- Implements the active messages model:
  - Nodes can exchange information by calling:
    - gasnet_AMRequestShort0(node, handler_idx). The message is sent and the corresponding function in the remote node is executed. Can call to: Reply().
  - Four type of messages:
    - `Short`
    - `Medium`
    - `Long`
    - *`LongAsync`*
  - All messages are synchronous (func returns when all params are copied and msg sent), except `LongAsync`

Why Gasnet?

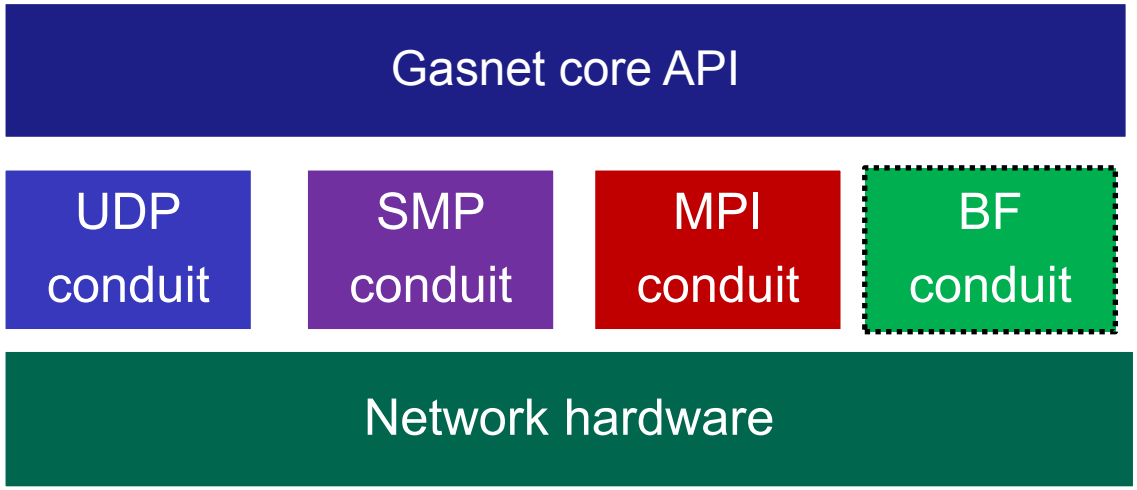
Gasnet is being used in Nanox in the current cluster implementation for parallel execution of tasks

Used too in:

- GCC/UPC Compiler
- Titanium Compiler
- Co-Array Fortran Compiler
- Chapel Project

Architecture of Gasnet

- Common API code, initialization and communication
- Specific conduit code for each network hardware
- Proposed a new conduit for Barrelfish, using flounder for communication



Gasnet and message passing in Barrelfish

Barrelfish

- Msgs sent to endpoints through a binding
- Remote funcs have to be registered with connect()/bind()
- Messages are completely **asynchronous**

Gasnet

- Msgs sent to nodes, no binding
- Remote funcs registered with gasnet_attach()
- Messages are sent **synchronously** except `LongAsync`

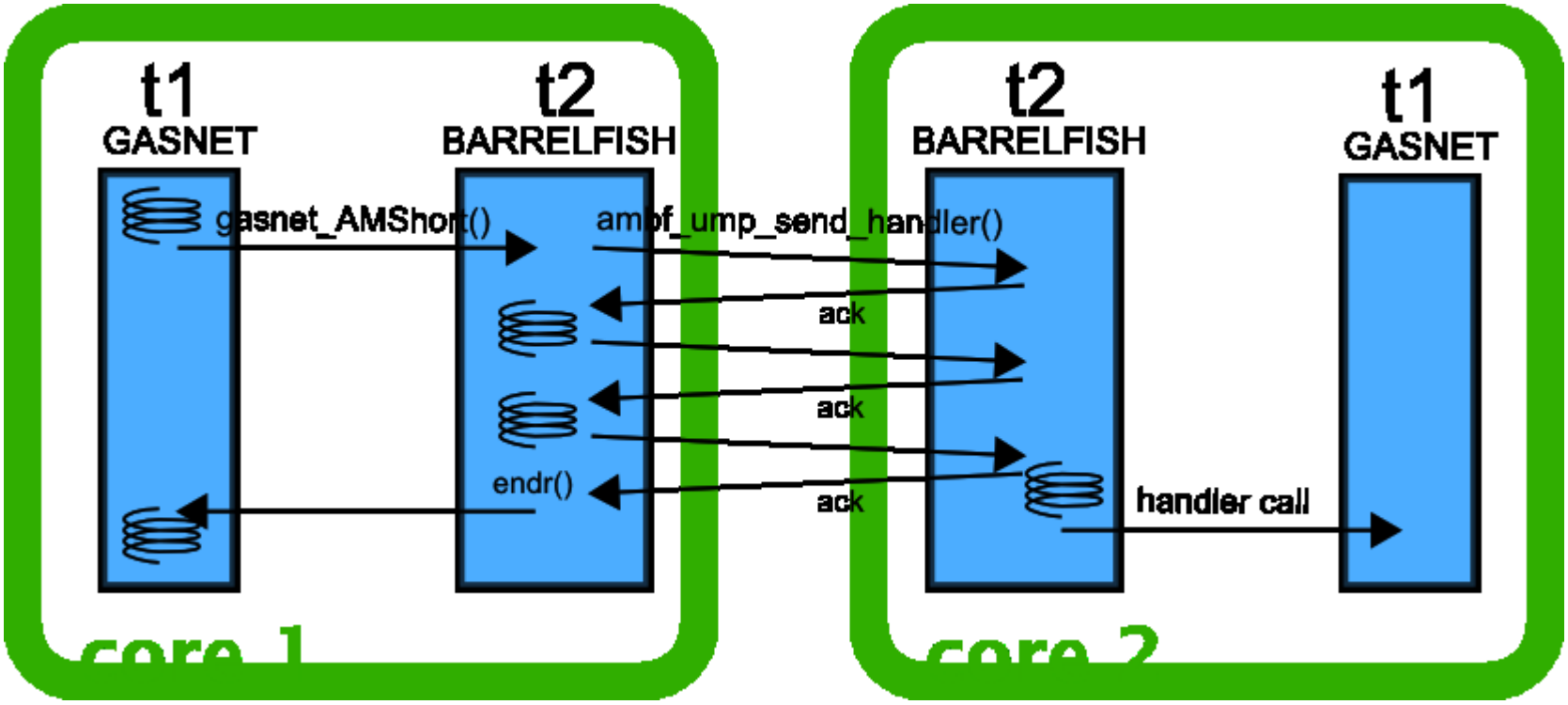
BF conduit implementation

Features required in the conduit:

- Create a complete mesh of connections among all nodes
- Central barriers for gasnet initialization
- Node gather to exchange information
- Event polling
- **Send/receive messages**

Gasnet BF conduit implementation

- Uses the Barrelfish flounder generated stub to pass messages
- To simulate the synchronous behaviour of gasnet, we use 2 threads.
- However, the binding cannot be handled by two threads concurrently without proper locking.



Gasnet BF conduit status

Currently stalled due to the memory leak, as gasnet benchmarks are transferring big amount of data: malloc()/free() continuously called.

Compiles and runs using the standard Barrelfish compiler, a script that calls the actual compiler

Expected to compile using the new GCC cross compiler

Questions?