

Heterogenous support for Treadmarks

Comp520 term project

Juan Navarro

Sameer Siruguri

Sitaram Iyer

Introduction

- Treadmarks DSM library: provides primitives for
 - ◇ allocation of shared memory
 - ◇ synchronisation
- Supports networks of homogeneous machines only
- Advantages of heterogeneity –
 - ◇ can obtain a larger number of machines by allowing machines of different architectures
 - ◇ people already have heterogeneous NOWs installed
 - ◇ will bring parallel computing to the masses

- Hence, we have extended the Treadmarks library to run Treadmarks applications unchanged in a heterogeneous environment.

The Issues

- Data transfers require appropriate conversion
 - ◇ endianness – big or little
 - ◇ alignments of data types in structs may be different
 - ◇ long variables are 8 bytes on Alpha/Ultra, and 4 bytes elsewhere
- The initial address for the shared memory will not be guaranteed to be the same across different architectures and/or OS'es
- Different address space sizes cause problems with location of the shared memory pages

- Unions – the particular type being accessed must be known at each program point that accesses the instance of the union

Overview of our work

- Modification of certain Treadmarks primitives appropriately to achieve the above
- Source translation: we concentrate on i386/sparc, so we implement
 - ◇ endianness conversions
 - ◇ struct realignment by padding
- Emphasized –
 - ◇ transparency
 - ◇ minimal performance degradation
 - ◇ extensibility

Data Conversion

- Each time a message is received
 - ◇ check architecture of sender
 - ◇ make all necessary data conversions
- Conversion of consistency information (e.g. vector timestamps) is straightforward
- For user data (pages or diffs) the exact layout or *data type map* of the page is required

Memory Allocation

- When allocating shared memory, the type of the data to be stored must be specified
- Restriction: a page contains only variables of the same type
- Modifications to Tmk_malloc
 - ◇ it takes a type-id as the second argument
 - ◇ it tags allocated pages with this type-id
 - ◇ minimum allocation unit is one page

Conversion of User Data

- Each time a page or page's diff is sent, the data type tag is attached
- The receiving side calls a conversion function that depends on the architecture of the sender, and on the data type
- Alternative
 - ◇ do the conversion at the sending side
 - ◇ save the converted diff in case it is later requested by other node from the same family of the last requester

Transparency

- These conversion functions are automatically generated in a preprocessing phase
- The appropriate second argument to `Tmk_malloc` is added in that phase

Other TreadMarks modifications

- At start-up:
 - ◇ negotiate logical page size
 - ◇ negotiate starting address of shared memory

Source translation

Parsing the application source code to

- Identify types of shared variables and pass `type_id` to `Tmk_malloc`
 - ◇ create a type – `type_id` mapping
- Change alignments and padding in structs to suit all machines
- Align arrays and structs to word boundaries
 - ◇ in addition to machine-specific alignment requirements
- Generation of conversion routines, one per machine, per type

Implementation

- SUIF compiler suite
- Phases – front end, various independent passes over intermediate code, back to C
- Our source translator is one stage in the SUIF compilation process

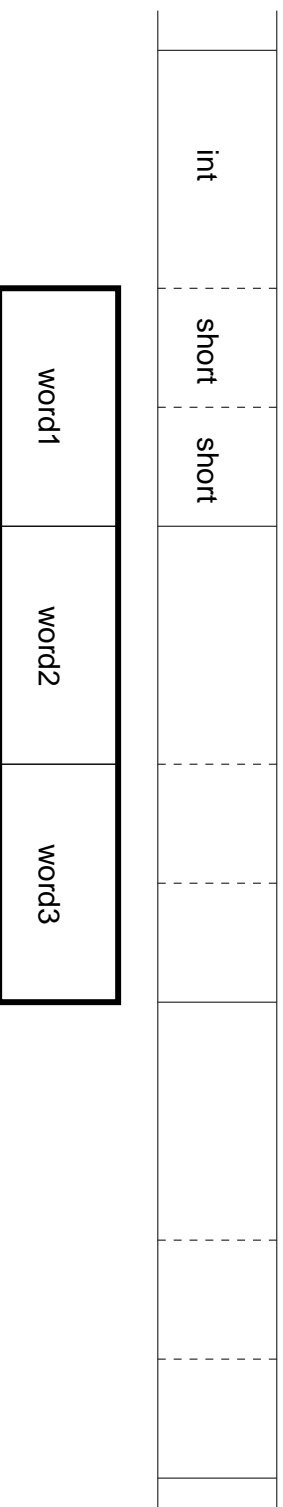
Conversion functions

- On the receiving end – depending on the data type and the sender architecture, appropriate conversion needs to be done.
- Conversion function: inputs – page, diff, type_id, machine_id action – convert the diff and apply it on the page
- Two options –
 - ◇ One function that uses generated type descriptions
 - ◇ Multiple generated functions

Workings of a conversion function

- The diff is a series of run-length encoded changes:
<offset into page, run-length>, new-word1, new-word2, ...
 The offset and run-length are word-aligned.
- Original diff_apply does a word-by-word copy
- Generated diff_convert_apply “knows” the type, so it maps words of the diff to it.

- Example: `struct { int x; short y; short z; }`



Performance

One diff_convert_apply

Original optimized diff_apply	466 clocks
diff_convert_apply without swabbing	514 clocks
diff_convert_apply with swabbing	1644 clocks

Overall application:

- Less than one second of difference in 45 seconds of execution

Conclusions

- Heterogeneous support for TreadMarks
- Marginal increase in total execution time
- Further extensions possible

Future Work

- Incorporating 32/64-bit address space conversion
- Certain assumptions about the data types
 - ◇ doubles on 64bit boundaries (not on x86)
 - ◇ arrays of doubles not used
- Allowing the use of unions