Pan Compiler

C. Loomis (LAL-Orsay)

Quattor Workshop (Bologna)
17-18 March 2008

Contents

- **Compiler Status**
- **Changes: v7 to v8:**
  - Language-level changes
  - Implementation changes
  - Functional changes
- **Migration Issues**
- **Discussion**
  - Feature requests, problems, etc.
  - Migration of standard templates
  - …

Compiler Status

- **Version 6 (deprecated)**
  - Old, c implementation of compiler
  - Frozen; no bug fixes or feature enhancements
- **Version 7 (production, current = 7.2.9)**
  - Production, java implementation of compiler
  - Bug fixes only (or very urgent functional changes)
- **Version 8 (development, trunk)**
  - Incompatible language-level changes compared to v6!
  - Development tags (v8.1.x) available for evaluation.
  - *Feedback from each site would help moving this to production.*

Language Changes

- **Removed features deprecated in v7:**
  - Keywords removed:
    - § define, delete, description, descro
  - Types removed:
    - § embed, fetch, stream

- **Newly deprecated features in v8:**
  - Bareword include:
    - § `include my/template;` ⇒ `include {'my/template'};`
  - Using 'type' for binding:
    - § `type 'my/path'=boolean;` ⇒ `bind 'my/path'=boolean;`
  - Lowercase automatic variables:
    - § loadpath, object, self, … -> LOADPATH, OBJECT, SELF, …

Language Changes

- **External path syntax**
  - Current path syntax (to be deprecated):
    - `//myobject/some/absolute/path`
  - Additional path syntax (preferred):
    - `my/object/tpl:/some/absolute/path`
    - Will (eventually) allow object templates to be namespaced!
- **Literal escaping of paths:**
  - Can escape part of a path:
    - `'/my/complex/{a/b}/path'⇒'/my/complex/a_2fb/path'`
    - Simpler than: `'/my/complex' = nlist(escape('a/b'), …;`
    - Perhaps can replace/simplify SPMA functions.
- **More limits on structure template content.**
  - Variables, function, and type statements not allowed.

Implementation Changes

- **Better handling of SELF**
  - Should be faster and less memory intensive.
  - More consistent in various contexts.
- **Some optimization**
  - Evaluation and use of compile-time expressions.
  - Specialized operators to avoid redundant checks at runtime.
  - Optimization allows stricter syntax checking in many cases.
- **“Read-only” resources**
  - Infrastructure in place to use read-only resources.
  - Intent is to avoid unnecessary copying of resources.
  - Not used yet; some semantic issues to work out.

- **GRIF full build: 472s (v7) ⇒ 357s (v8); ~25% faster.**

New/Changed Functions

- **`format()`**
  - Printf-like capabilities (formatting of numbers, strings, etc.)
  - Can avoid incremental building of configuration file contents
- **`is_defined(a), is_boolean(a), ...`**
  - All return 'false' and not error if variable a does not exist.
  - `(exists(x) && is_defined(x))` ⇒ `(is_defined(x))`
- **String manipulation**
  - `to lowercase(), to uppercase()`
  - `split()`
  - `replace()`
- **`to_string()`**
  - Will accept any element, resources included!

Misc. Functionality

- **New automatic variable: TEMPLATE**
  - Gives name of template that initiated the DML block.
  - Does not change with function calls (including `create()`).
  - Is this what is desired?

- **New output format:**
  - Write machine template as a dot (Graphviz) file.

- **(Proto-) Documentation:**
  - Pan compiler and language manuals (PDF, HTML).
  - Pan tutorial (PDF, HTML).
  - README (PDF, HTML).
  - Man pages for panc, scripts, and built-in functions.

Logging

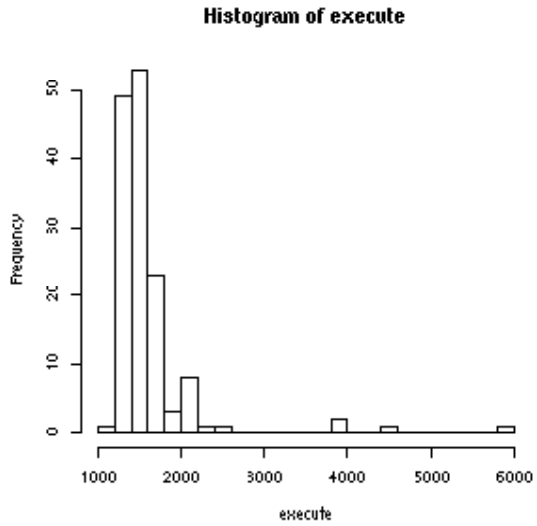
- **Logging capabilities added:**
  - Logging types: “task”, “call”, “memory”, “all”, “none”.
  - Must set the logging file to use for output.
  - Message are short and easily parsed for analysis.
  - Logging is global in the JVM; watch out for interference!
  - *Cost: ~15% slower (all logging), ~75 MB file for 147 profiles.*

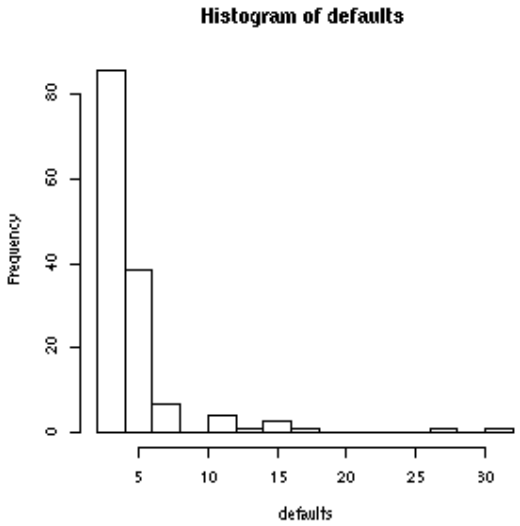
- **Example analysis scripts for:**
  - Memory usage vs. time.
  - Tasks per thread vs. time.
  - Graph of call (include) structure.
  - Performance studies (both top-down and bottom-up).

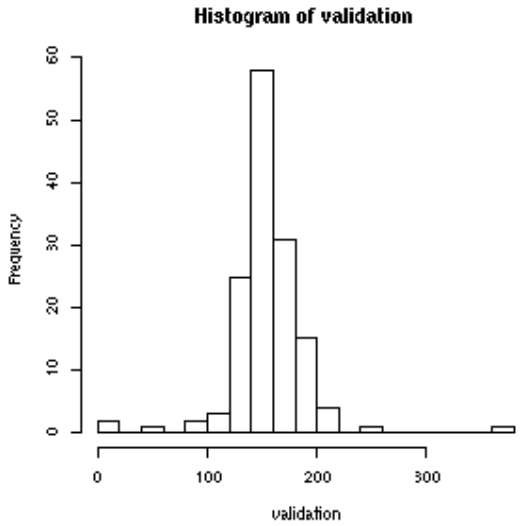
Logging

Distributions of execution, insertion of defaults, and validation.

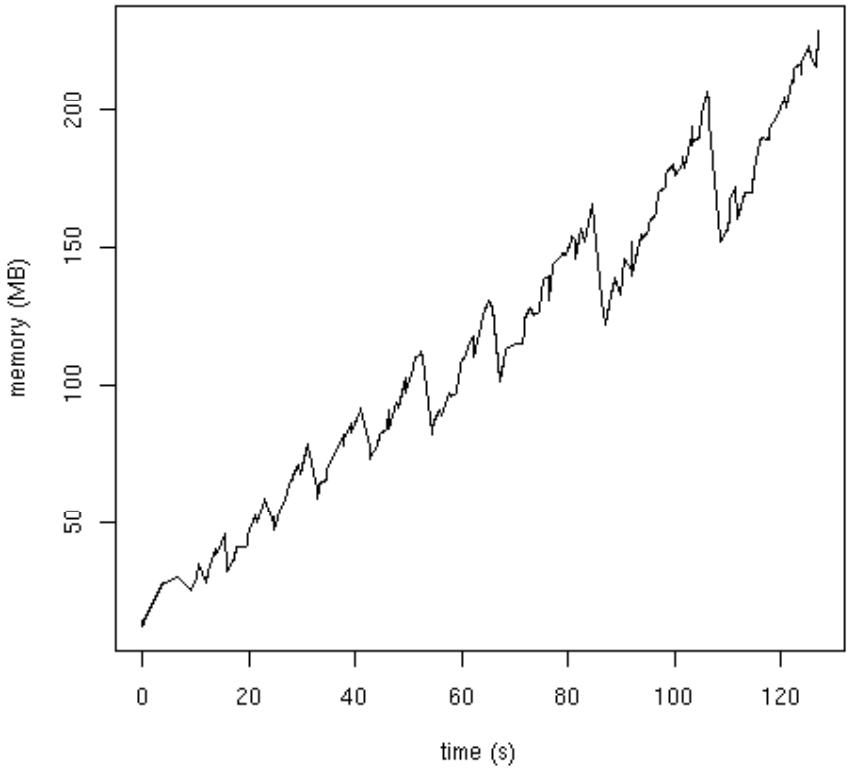
Typical times:
* execution: ~1400 ms
* defaults: 2 ms
* validation: 160 ms

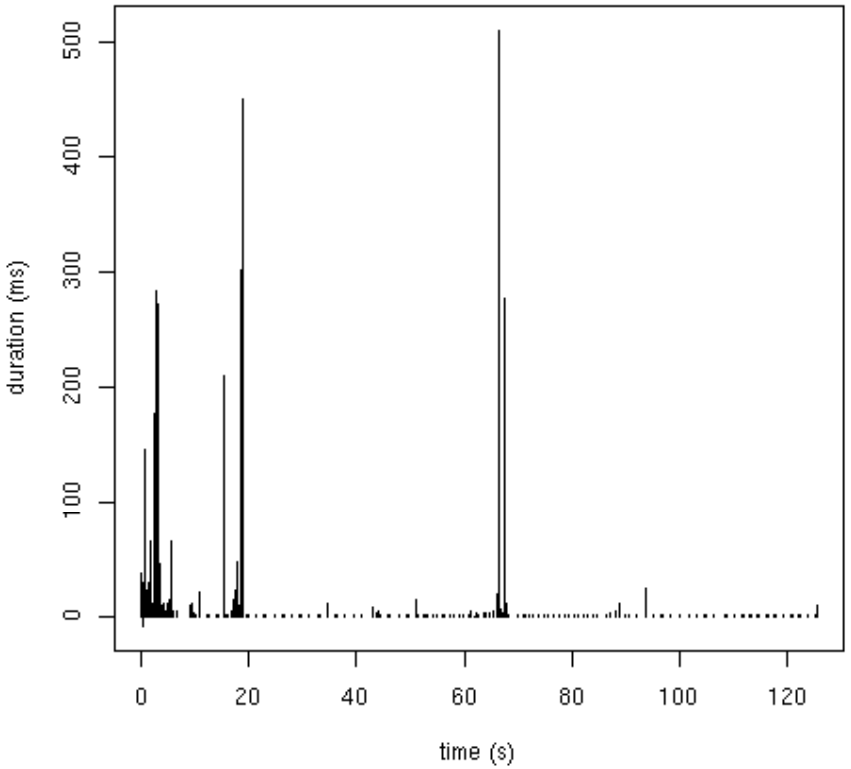






Logging

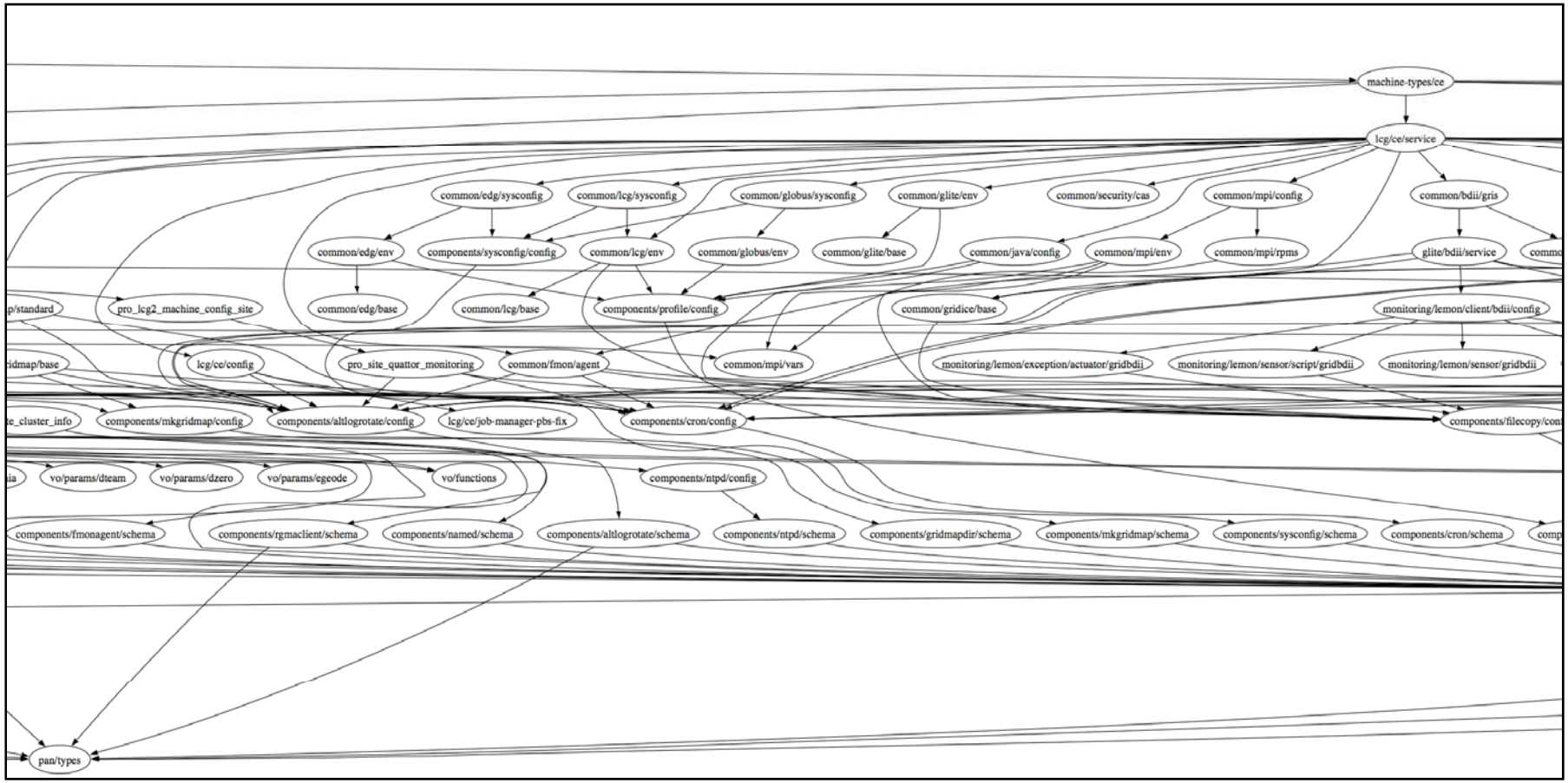




Memory and compilation histories.

Logging/Includes

Complex inclusion graph!
Can customize information or format.



Logging/Profiling

```
4082 ms | pro_lcg2_config_site_maui
 262 ms | profile_grid10
 181 ms | vo/init
 165 ms | quattor/repository_cleanup
  89 ms | lcg/ce/service
  69 ms | rpms/printing
  56 ms | common/gip/lcg-ce
  55 ms | rpms/compat_arch_support
  55 ms | config/glite/3.1/base
  47 ms | repository/config/os
```

Bottom-up view shows "hot" templates.

```
profile_grid10 (6216 ms)
|--machine-types/ce (5710 ms)
|   |--machine-types/base (5019 ms)
|   |   |--pro_site_databases (0 ms)
|   |   |--pro_site_global_variables (28 ms)
|   |   |   |--pan/functions (1 ms)
|   |   |   |   |--pan/types (0 ms)
|   |   |   |   |--quattor/functions/network (0 ms)
|   |   |   |       |--pan/types (0 ms)
|   |   |--pro_site_functions (0 ms)
|   |   |--quattor/profile_base (15 ms)
```

Top-down view shows "hot" regions in call stack.

Migration Issue(s)

- **Change in global variable handling:**
  - v7: variable X = exists(X) ⇒ true or false
  - v8: variable X = exists(X) ⇒ always true

- **Consequences for 'tri-state' variables in QWG:**
  - Should use (null, true, false) in preference to (undef, true, false).
  - Changes are/will be incorporated into next QWG release.

```
variable X ?=
  if (!exists(X)) {
    undef;
  } else {
    false;
  };
```

```
variable EXISTS = exists(X);

variable X ?=
  if (!EXISTS) {
    undef;
  } else {
    false;
  };
```

Discussion

- **Migration of standard templates**
  - Avoid deprecation warnings when using v8.
  - Use of more modern features of compiler.
- **Authorization/Entitlements**
  - What change?
  - Who did it?
  - Was it authorized?
- **Feature requests, problems, etc.**
  - Who's using what options of panc script?
  - Urgent features that aren't being addressed?
  - Specific problems?

- **Other topics related to pan or pan compiler?**