Pan Compiler

C. Loomis (LAL-Orsay)

Quattor Workshop (Madrid)
29-30 October 2007

Contents

- Current Status
- Planned Language Changes
- Performance
- Proposed Feature Changes
- Roadmap Discussion

Links

- **Pan Documentation**
  - Language (https://trac.lal.in2p3.fr/LCGQWG/wiki/Doc/panc)
  - Compiler (https://trac.lal.in2p3.fr/LCGQWG/wiki/Doc/compiler)

- **Packages & Installation**
  - https://trac.lal.in2p3.fr/LCGQWG/wiki/Download/panc
  - QWG template distribution
  - Quattor distribution
  - ETICS repository

- **Development**
  - Roadmap (https://trac.lal.in2p3.fr/LCGQWG/roadmap)
  - Bugs (https://trac.lal.in2p3.fr/LCGQWG/report)

Current Status

- **Version 6 (deprecated)**
  - Old, c implementation of compiler
  - Frozen; no bug fixes or feature enhancements
  - Used by default for CDB
- **Version 7 (production, current = 7.2.6)**
  - First, production java implementation of compiler
  - Bug fixes being applied; (very) limited feature enhancements
  - Almost 100% backward-compatible with Version 6
  - Used by default for SCDB
- **Version 8 (development, trunk)**
  - Incompatible language-level changes compared to Version 6
  - Expect to have production version ready before Christmas

Performance

- **Slow compilation of (CERN-like) profiles is the issue that has kept the java-implementation from being universally adopted.**

- **To investigate:**
  - Added rudimentary profiling to v7.2 branch.
  - Developed set of performance tests.

- **Test configuration:**
  - Pan compilers: v6.0.4, v7.2.5, and v7.2.6
  - Machine: Xeon (64bit), 2.3 GHz, 8 (2x4) cores, 16 GB RAM
  - OS: SL4, 2.6.9 kernel
  - Java: JDK 1.6.0-01, -Xmx8192M

Generic Performance Tests

Time in milliseconds (ms) to complete tests. Smaller number is faster compilation.

	v6.0.4 (c)	v7.2.5 (java)		v7.2.6 (java)	
	mono	mono	multi	mono	multi
assign	12880	2826	3923	2568	3978
cond-assign	13002	1827	1566	1406	2279
include	12871	758	1285	744	1726
cond-include	13004	1323	1655	1369	2210
delete	13137	2951	1683	2013	1537
f-noargs	13186	1993	1761	1689	2515
f-1arg	13519	2804	2433	2365	2927
var-self-ref	2205	83818	27307	2717	2034

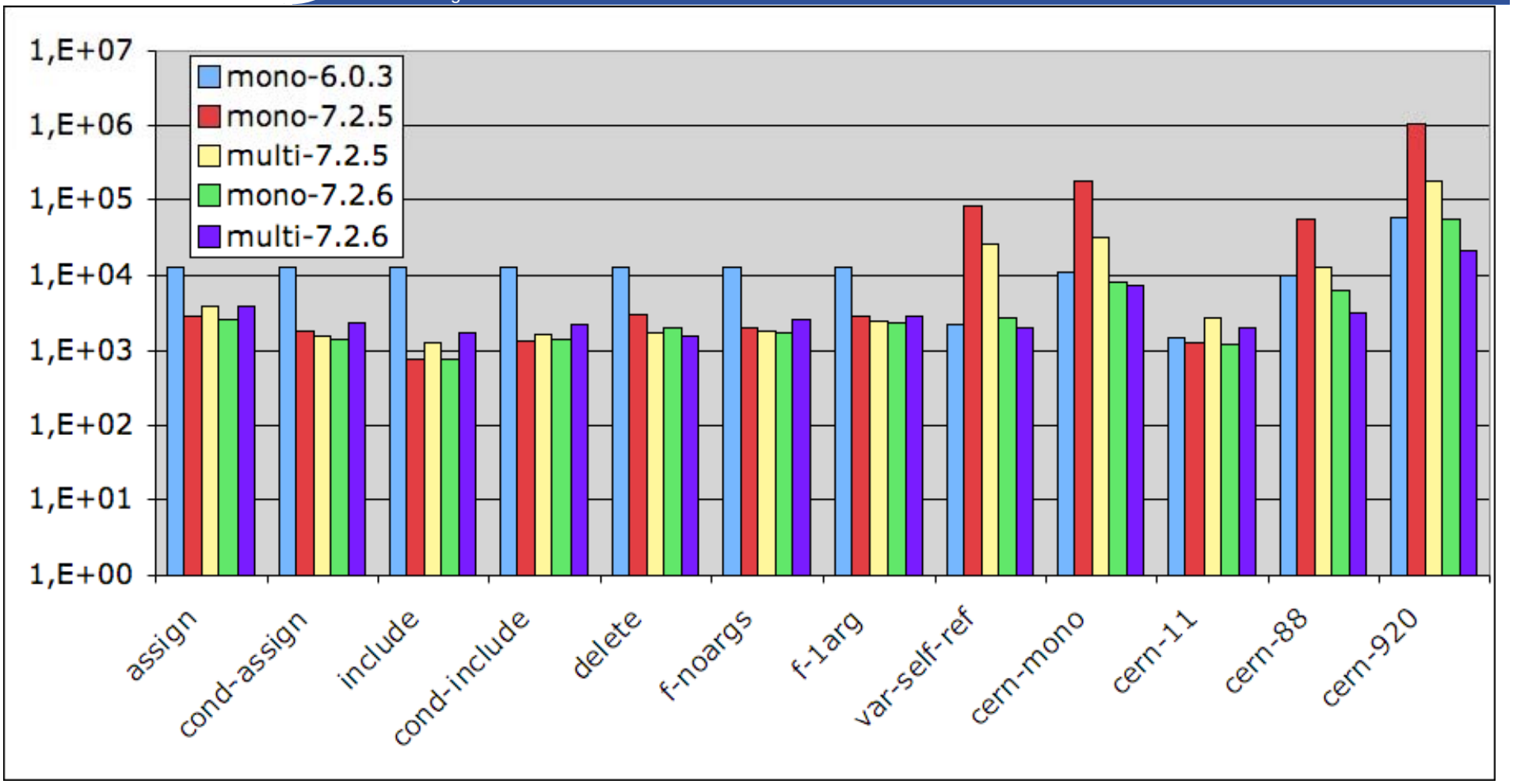
Performance for CERN Profiles

Time in milliseconds (ms) to complete tests. Smaller number is faster compilation.

Memory use around 2 GB for 920 profile test with java version.

	v6.0.4 (c)	v7.2.5 (java)		v7.2.6 (java)	
	mono	mono	multi	mono	multi
cern-mono (100)	11290	173690	33233	8367	7530
cern-11	1509	1273	2704	1170	2022
cern-88	10232	57170	12896	6437	3110
cern-920	61574	1057519	174377	56460	22240

Performance



Time in milliseconds (ms) to complete tests. Smaller number is faster compilation.

Planned Changes

Purpose of language changes is to simplify the pan grammar and speed-up the compilation; other changes, to enhance functionality.

	enhancements	deprecated	unsupported
v7	foreach stmt. 'bind' statement bit & misc. functions optional gzipped profiles rudimentary logging	'define' keyword 'description' keyword 'descro' keyword 'delete' stmt.	
v8	'format' function primitive type conflict err. i18n support perf. and include logging auto. var. for tpl. name 'final' for structure tpls. simple DML optim. 'copy-on-write' optim.	lowercase auto. vars. 'type' synonym for 'bind'	'define' keyword 'description' keyword 'descro' keyword 'delete' stmt.
v9		include my/tpl;	lowercase auto. vars. 'type' synonym for 'bind'
v10			include my/tpl;
v11	include DML w/o braces		

Cast Operator

- **https://trac.lal.in2p3.fr/LCGQWG/ticket/74**

I have thought of introducing a cast operator into pan. However, it would be more of a replacement for the to_string(), etc. functions. Something like:

variable x = (string) 3;

However this does indeed cause several complications. The easiest to resolve is just the changes to the grammar. Although these changes would be extensive, I think that they could be done. A more serious complication is a semantic one. What would the following mean?

variable x = (mytype) list('a','b');

Would casting be allowed for user-defined types, and if so, does this imply validation at the point of the cast? This would be useful to find errors as early as possible in the build process.

In the end though, I think the complications outweigh any benefit at this point. We might bring this back for a release beyond v10.

Forced Assignment

- https://trac.lal.in2p3.fr/LCGQWG/ticket/75

What you're actually asking for though is not really a cast operator but something more like a forced assignment. Something like:

variable x = 47; variable x := list(foo);

that will do the equivalent of:

variable x = 47; variable x = undef; variable x = list(foo);

and hence avoids the check on the primitive types when an assignment is made.

I must say I have mixed feelings about this because I think this check more often than not catches real mistakes. But as you say, it would be useful when dealing with the QWG distribution.

Roadmap Discussion

- **Compiler features**
  - Are the planned features for each release OK?
  - Are there new features that aren't in plan?

- **Adopting new panc releases**
  - Who will verify that panc v7+ works with CDB?
  - On what timescale can we drop panc v6 support?
  - What time interval between major panc releases?