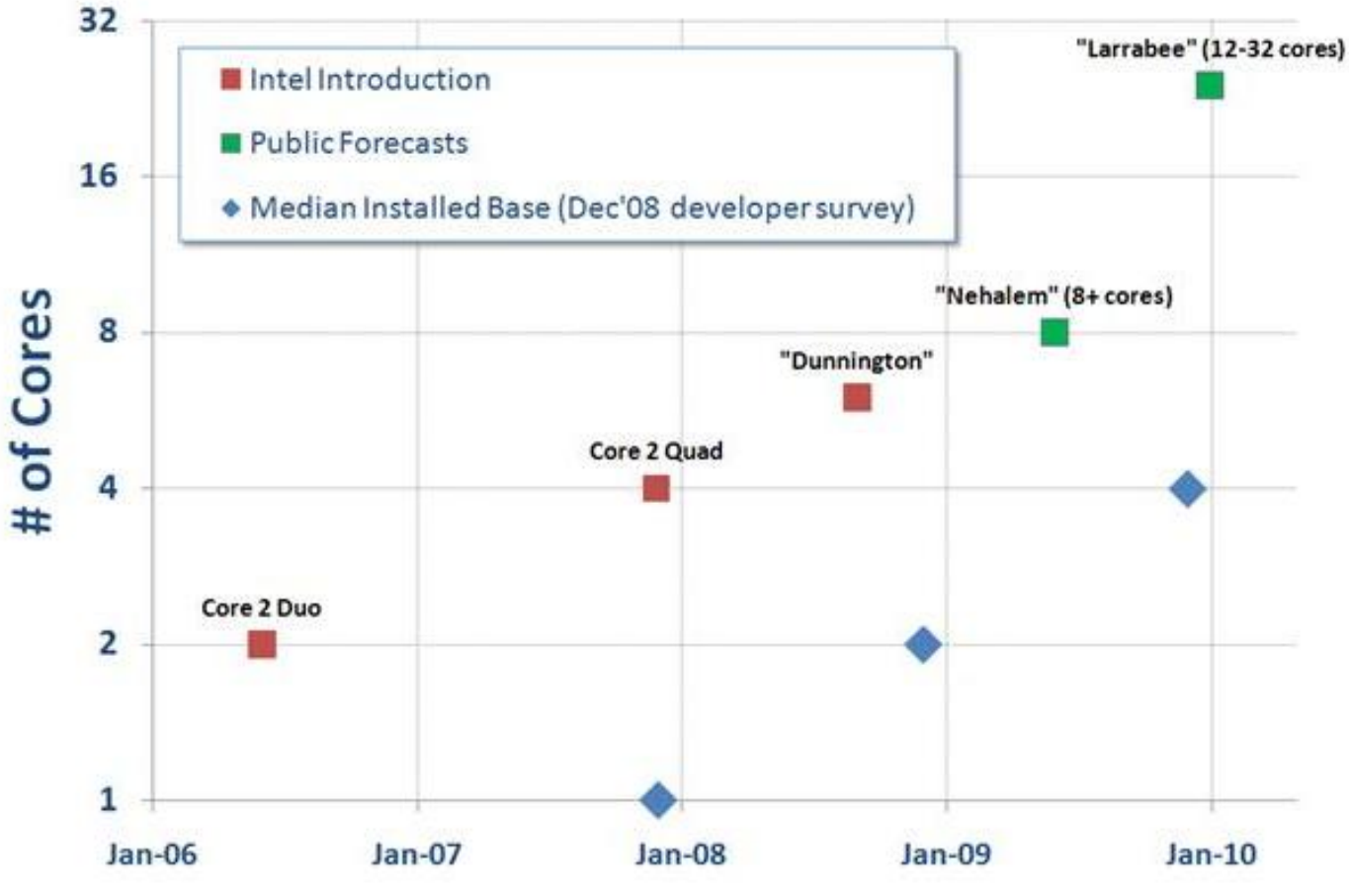
Algorithm Configuration for Portfolio-based Parallel SAT-Solving

Holger Hoos[1], Kevin Leyton-Brown[1],
Torsten Schaub[2], Marius Schneider[2]

[1] Department of Computer Science
University of British Columbia
Vancouver, BC, Canada

[2]Institute of Computer Science
University of Potsdam
Potsdam, Germany

The rise of main-stream multi-core computing



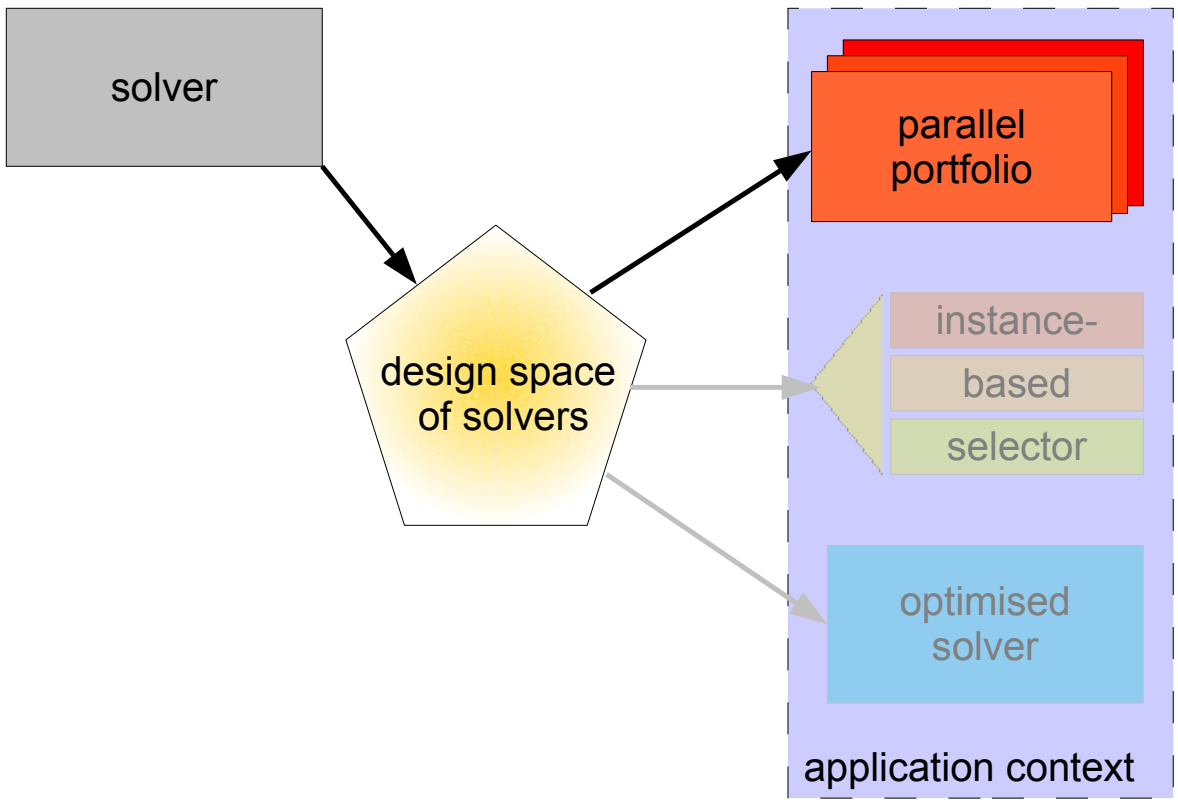
`http://software.intel.com/en-us/articles/gaming-into-the-future`

Parallel SAT Solvers

2008 SAT RACE	*2009 SAT Comp.*	*2010 SAT RACE*	*2011 SAT Comp.*
ManySat	gNovelty+-T	antom	borg-sat
MiraXT	ManySAT	ManySAT	clasp
pMiniSat	satake	plingeling	csls
	ttsth	PMiniSAT	CryptoMiniSAT
		SArTagnan	Plingeling
			ppfolio

- (Slowly) increasing number of parallel SAT solvers
- Designing effective parallel solvers is challenging!

Key idea: Automatically generate parallel solver
from highly parametric sequential design / sources



Parallel SAT Solving

- Search space splitting
  e.g., `clasp` with guiding path
- Competitive parallel portfolio with clause sharing
  e.g., `Plingeling` and `CryptoMiniSat`
- Competitive parallel portfolio w/o clause sharing
  e.g., `Plingeling` and `CryptoMiniSat`

Ingredients for parallel SAT solver

based on competitive parallel portfolio

- Parametric solver A
- Configuration space C
- Instance set I
- Algorithm configurator AC

That's all!

Recipe for parallel SAT solver

based on competitive parallel portfolio

1. Use algorithm configurator to produce multiple configurations of given solver that work well together
2. Run configurations in parallel until one solves given instance

Fully automatic method!

Recipe: GLOBAL

for parallel SAT solver based on competitive parallel portfolio

- For k portfolio components (= processors/threads), consider combined configuration space C^k of k copies of given parametric solver
- Use configurator AC to find good joint configuration in C^k (standard protocol for current configurators: pick best result from multiple independent runs)
- Configurations are assessed using (training) instance set I

Challenge: Large configuration spaces (exponential in k)

Recipe: GREEDY

for parallel SAT solver based on competitive parallel portfolio

- Add portfolio components, one at a time,
  starting from single solver
- *Iteration 1:* Configure given solver A using configurator AC
  $\rightsquigarrow$ single-component portfolio A^1
- *Iteration* $j = 2 \ldots k$*:* Configure given solver A using AC
  to achieve optimised performance of
  extended portfolio $A^j := A^{j-1} \,||\, A$
  i.e., optimise improvement in A^j over A^{j-1}

Note: Similar idea to many greedy constructive algorithms
(including `Hydra`, Xu *et al.* 2010)

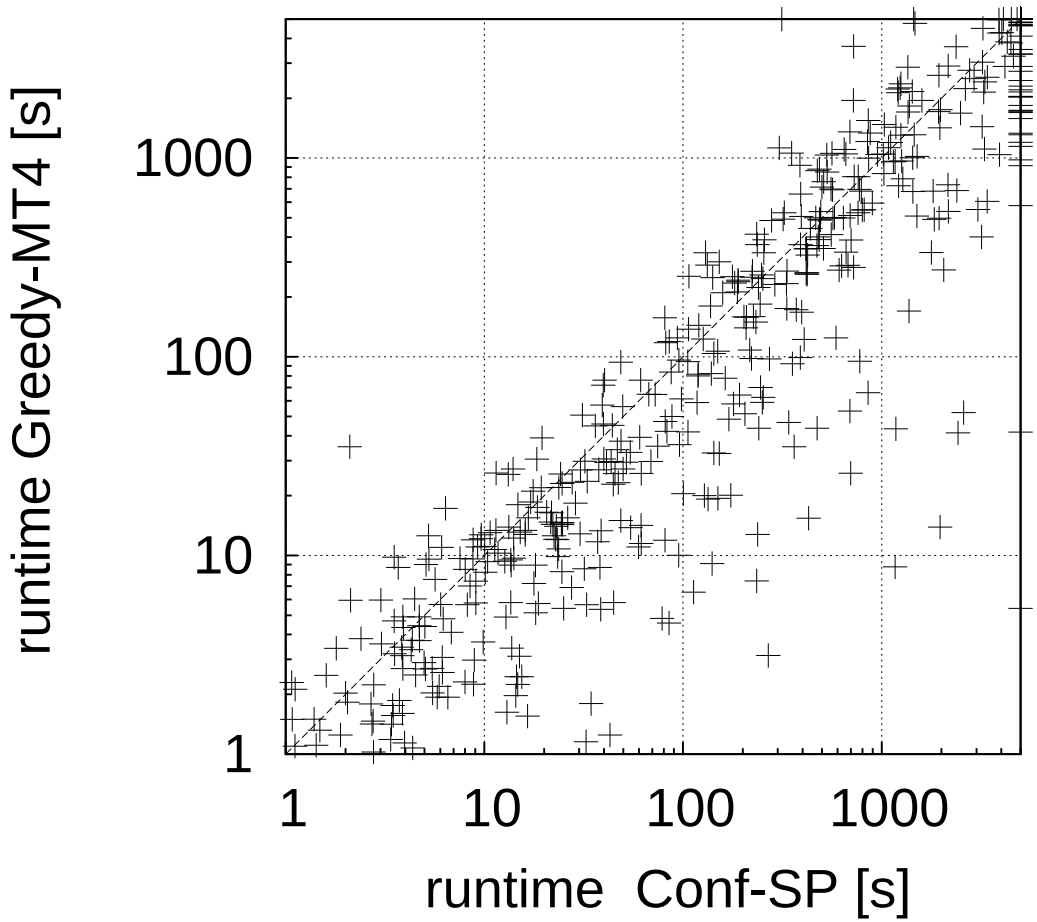
Product: parallel *Lingeling* (v.276)

on *SAT Comp. Application* instances

	PAR10	Overall Speedup *vs* Configured-SP	Avg. Speedup *vs* Configured-SP
Default-SP	3747	0.93	1.44
Configured-SP	3499	1.00	1.00
Plingeling	3066	1.14	7.39
Global-MP4	2734	1.27	10.47
Greedy-MP4	1341	2.61	3.52

Product: parallel *Lingeling* (v.276)

on *SAT Comp. Application* instances



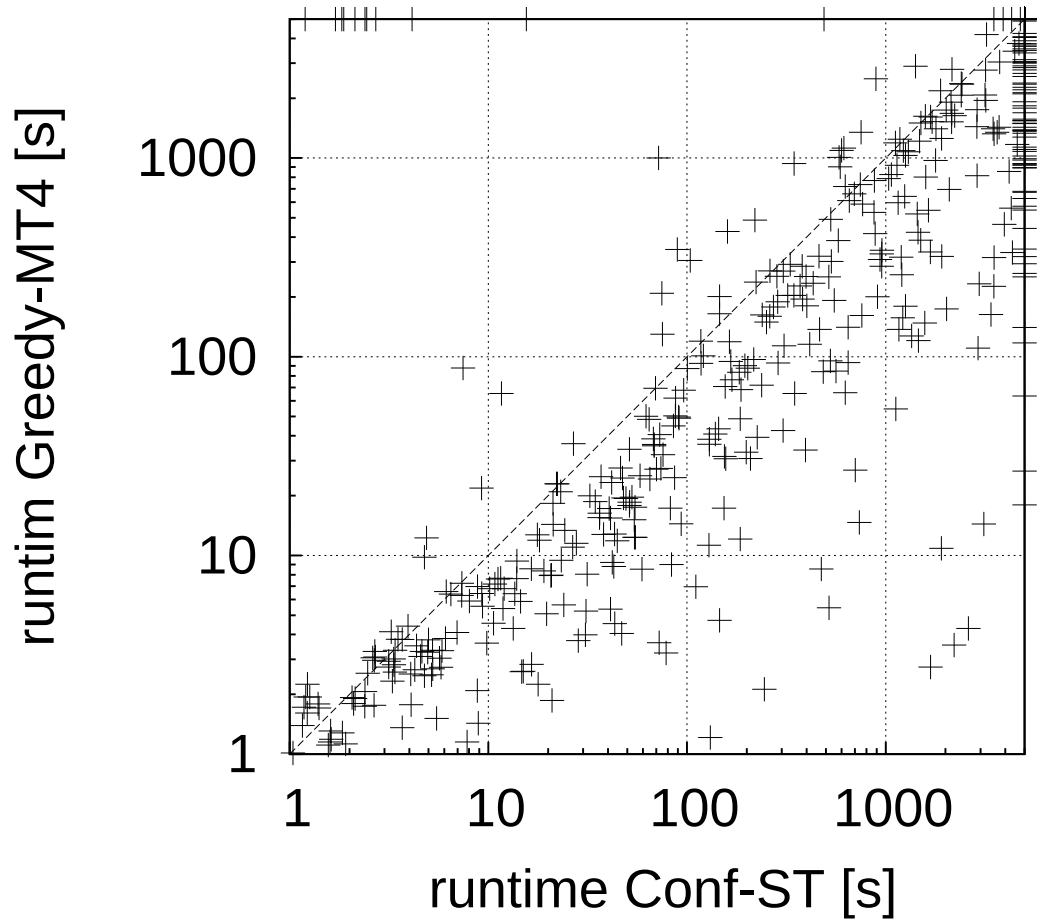
Product: parallel *clasp* (v.2.0.2)

on *SAT Comp. Application* instances

	PAR10	Overall Speedup *vs* Configured-SP	Avg. Speedup *vs* Configured-SP
Default-SP	7560	0.82	4.46
Configured-SP	6170	1.00	1.00
Default-MP4	2324	2.65	7.58
Global-MP4	3604	1.71	6.36
Greedy-MP4	2277	2.71	9.47

Product: parallel *clasp* (v.2.0.2)

on *SAT Comp. Application* instances



Conclusions

- Parallel SAT solving is gaining importance
- Designing high-performance parallel SAT solvers can be challenging
- Fully automatic method:
  1. Use algorithm configurator to produce multiple configurations of given solver that work well together
  2. Run configurations in parallel until one solves given instance
- At least as good as hand-crafted parallel SAT solvers

Future Work

- Parallel portfolios with different solvers
- New configuration protocols / configurators
- Generic mechanisms for cooperation between portfolio members