2nd International Workshop on CPAchecker (CPA'17)
Paderborn, Germany
September 4, 2017

Combinations of Methods for Efficient Software Model Checking

Vitaly Mordan
mordan@ispras.ru

Institute for System Programming of the Russian Academy of Sciences

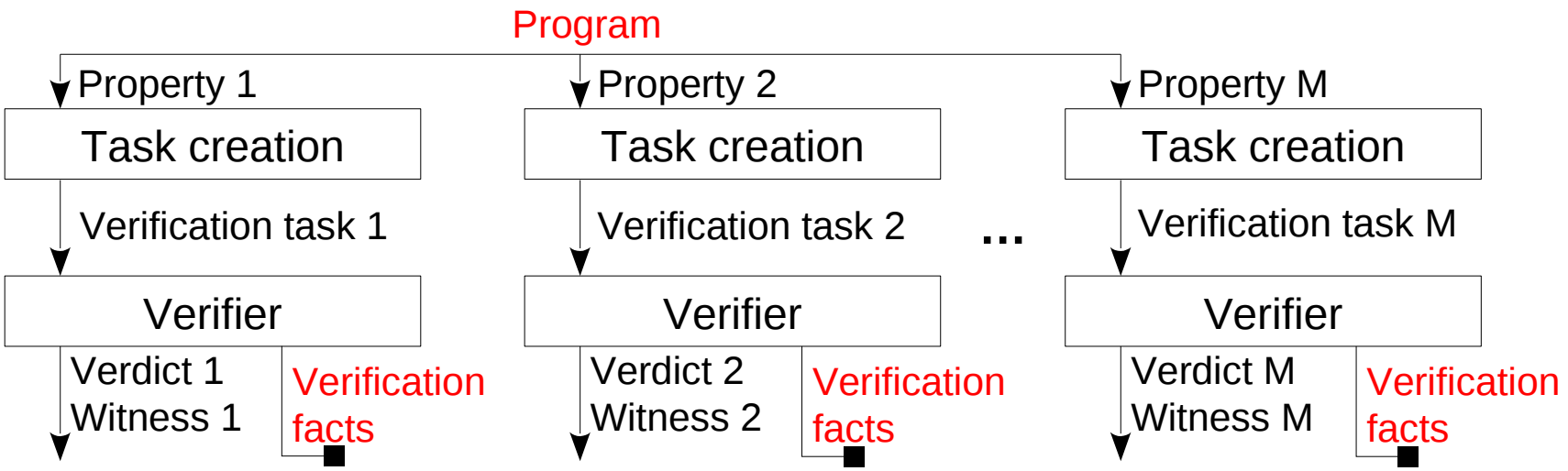
Software Model Checking

- SV-COMP
  - N verification tasks (C program + property)
  - **Required resources**: ~N ~490 days of CPU time*
- Real Software
  - N programs
  - M properties → Multi-property verification
  - K versions → Regression verification
  - **Required resources**: ~N***M*K**

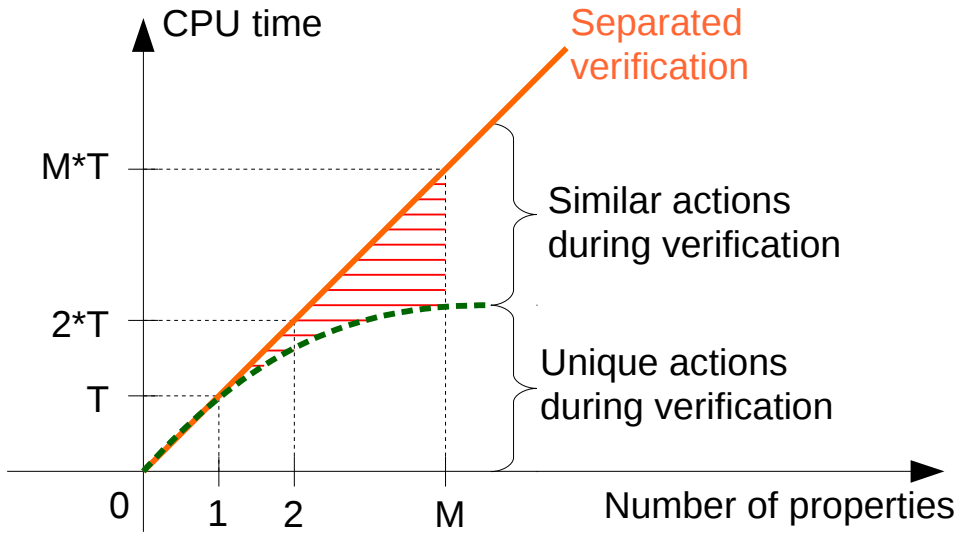
Combination of methods → Increase efficiency

* D. Beyer. Software Verification with Validation of Results (Report on SV-COMP 2017). TACAS 2017.

Separated Verification (Basic Method)



Verification facts* are lost after a verifier run

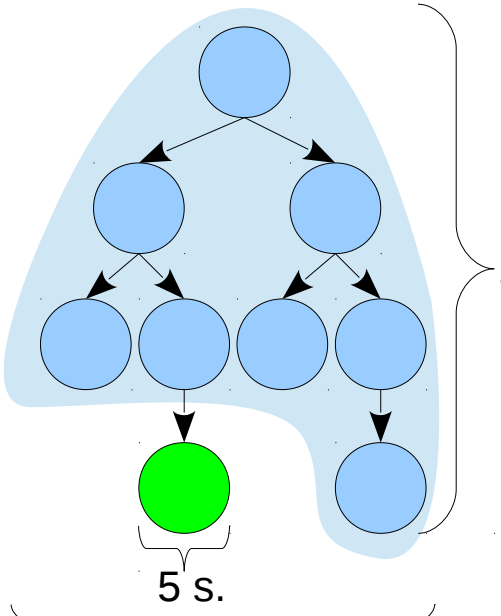


* Such as abstraction precision, abstract reachability graph, witness, etc.

Multi-Property Verification by Example

Separated verification

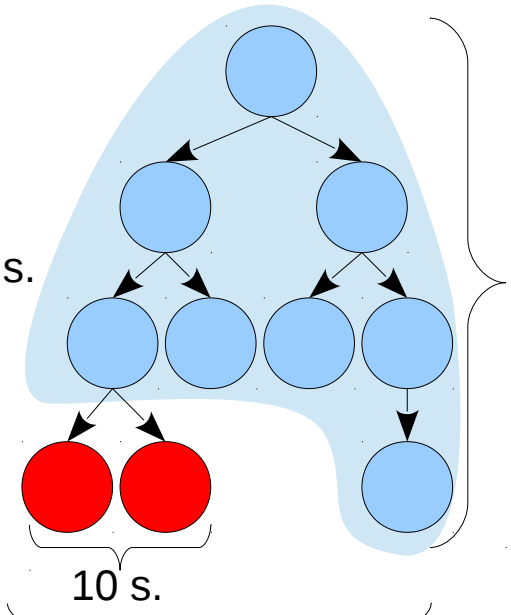
Property 1



105 s.

+

Property 2

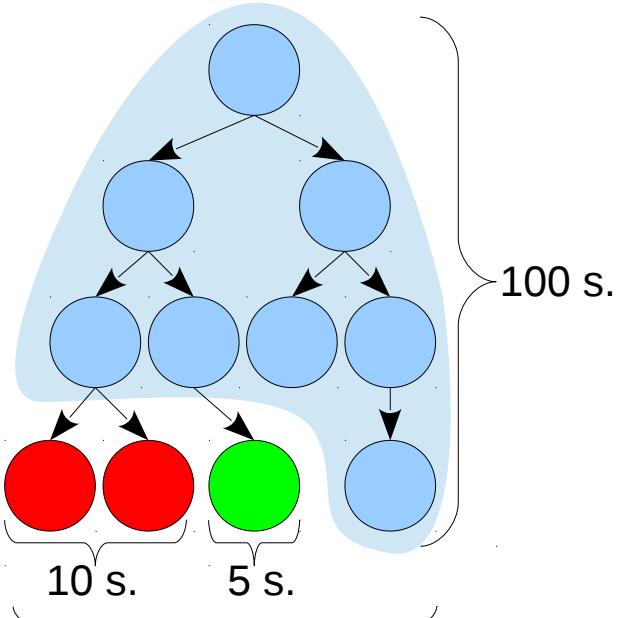


110 s.

= **215 s.**

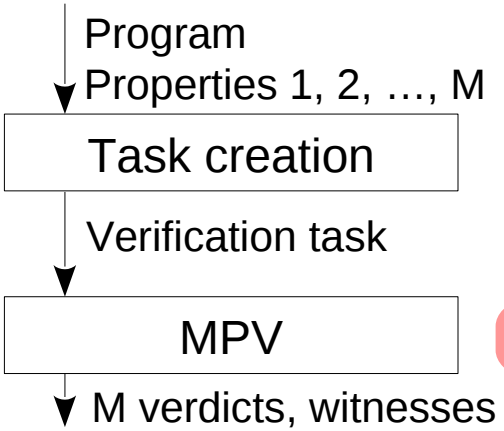
Combined verification

Properties 1 and 2



115 s.

Multi-Property Verification



Warning: state space explosions!

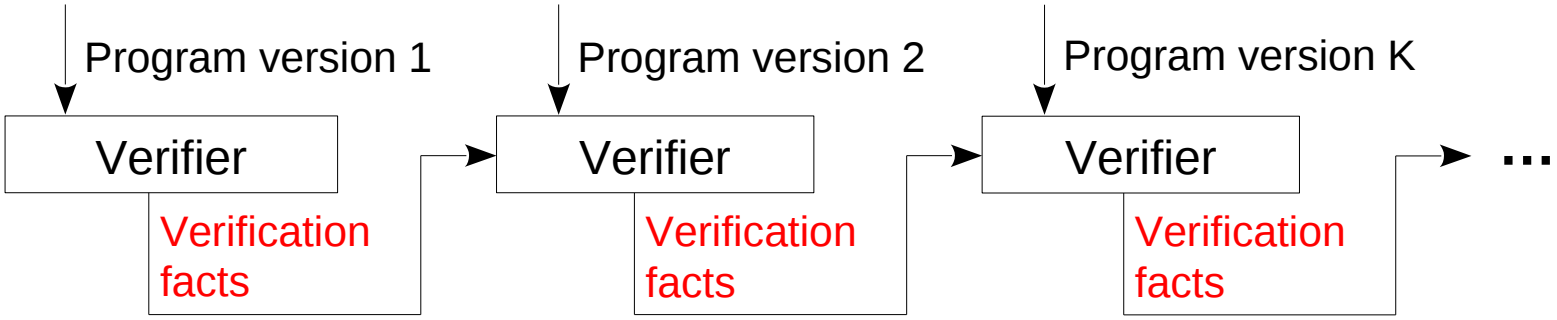
- Conditional Multi-Aspect Verification (CMAV*)
  - Limit resources for checking of each property
- Specification DeComposition (SDC**)
  - Decompose all properties on smaller sets

➡ Increase of efficiency in several times

* V. Mordan, V. Mutilin. Checking Several Requirements at once by CEGAR. LNCS 9609.

** S. Apel, D. Beyer, V. Mordan, V. Mutilin, A. Stahlbauer. On-The-Fly Decomposition of Specifications in Software Model Checking. FSE 2016.

Regression Verification*



- Save verification facts on initial version
- Use it for verification of next versions

➡ Increase of efficiency in several times

* D. Beyer, P. Wendler. Reuse of verification results. LNCS 7976.

The Suggested Combinations

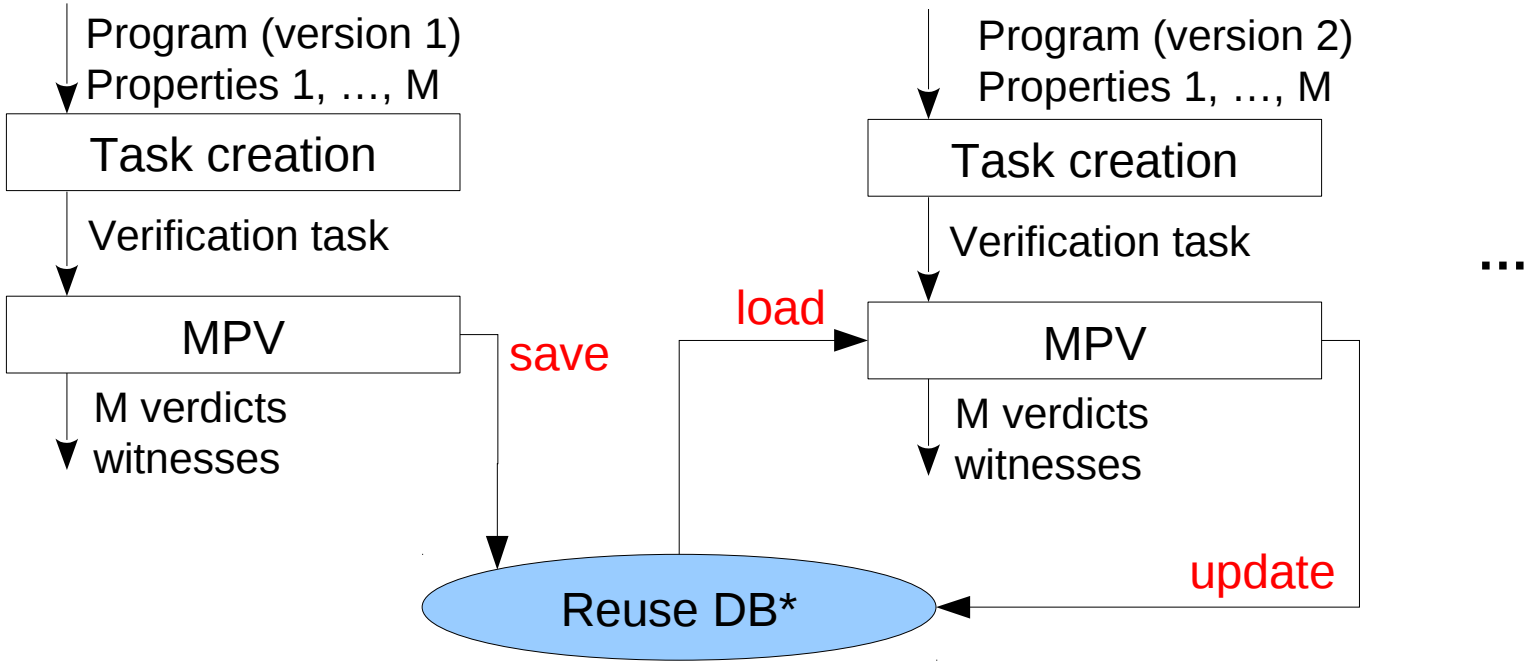
Main idea of combinations:
Regression Multi-Property Verification (RMPV)

RMPV-1) Multi-property verification + regression verification

RMPV-2) Combination of different multi-property verification methods

RMPV-3) Combination of RMPV-1 and RMPV-2

Multi-Property Verification + Regression Verification (RMPV-1)



➡ Increase efficiency for the next versions

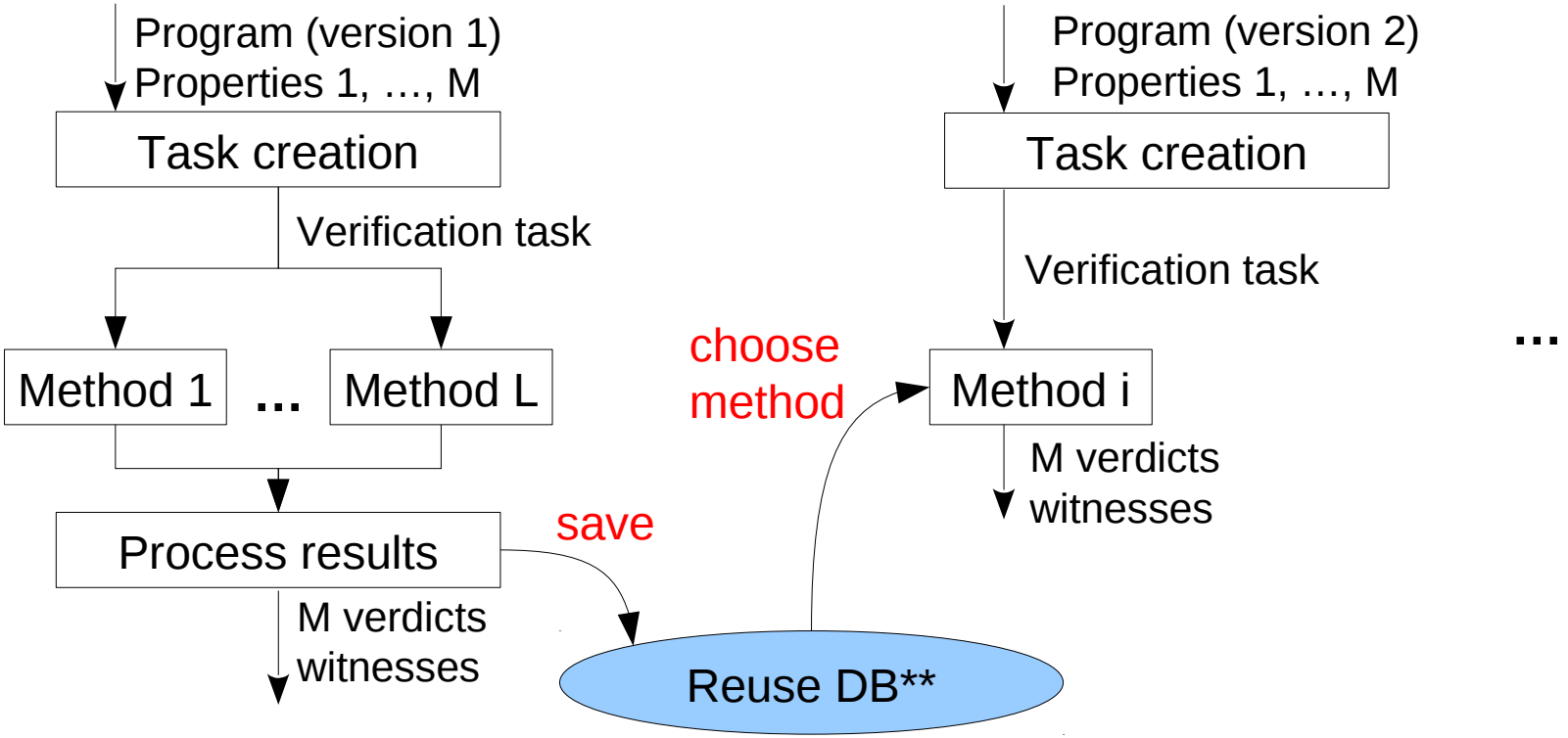
* Contains entries <verification task>: <verification facts>. Verification task if determined by corresponding program and properties composition.

Mitigate Negatives (RMPV-1 + MN)

- State space explosion problem
  - Property was not successfully verified
  - Reuse of verification facts may lead to state space explosion
- Reuse verification facts **only** for successfully checked properties

➡ *Priority*: increase efficiency rather than try to solve new tasks (may lead to state space explosion)

Combination of Different Multi-Property Verification Methods* (RMPV-2)

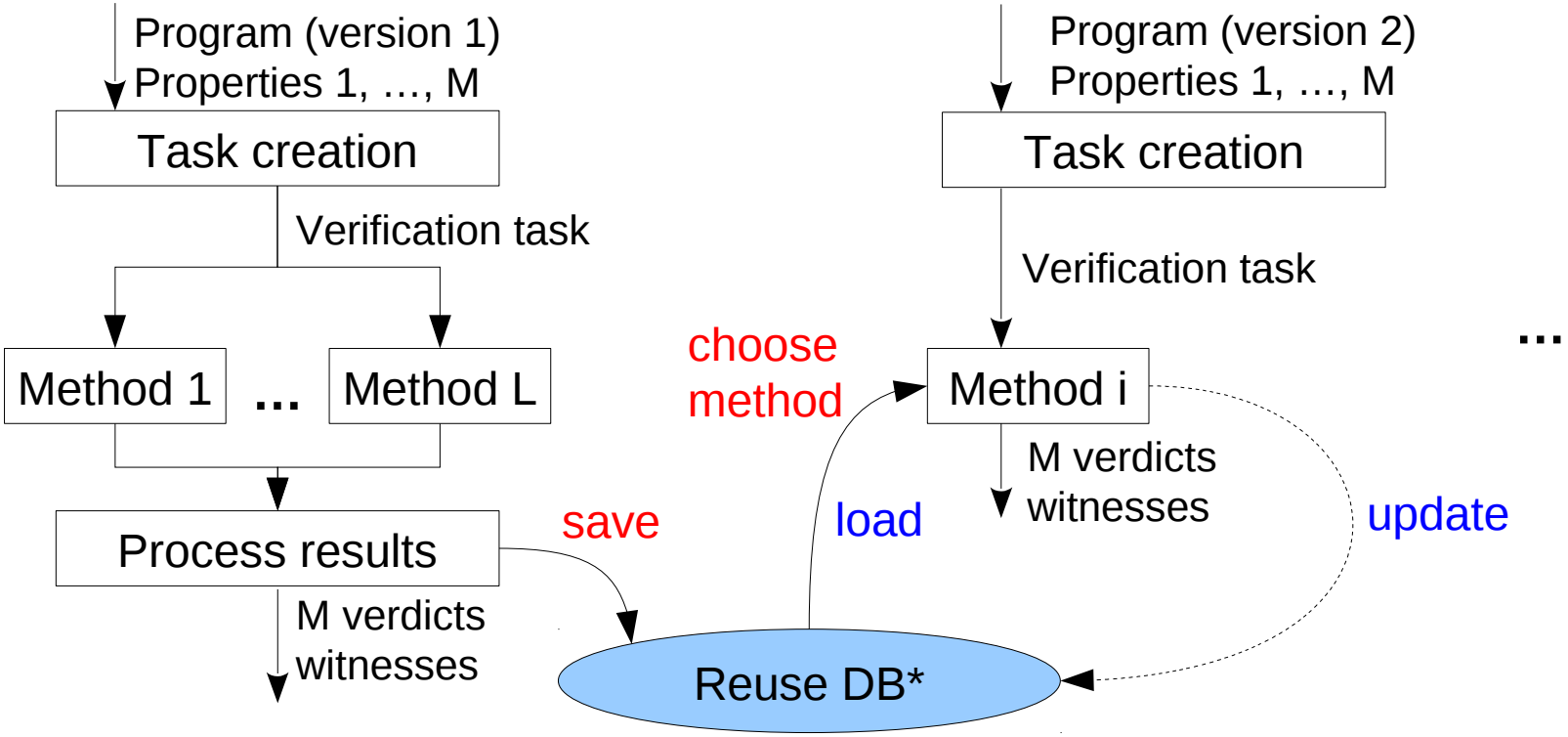


Cheap way to use strengths and minimize weaknesses

* Each MPV method can be implemented in a separated verifier.

** Contains entries <verification task>: <method>.

Combination of RMPV-1 and RMPV-2 (RMPV-3)



➡ Unites strengths of RMPV-1 and RMPV-2

* Contains entries <verification task>: <verification facts>, <method>.

Implementation

- Verification task creation
  - LDV Tools (verifies Linux kernel modules)
- Multi-property verification methods
  - CMAV (CPAchecker)
  - SDC (CPAchecker)
- Regression Verification
  - Precision reuse* (value and predicate)
- Benchmarking
  - BenchExec**

* Beyer D., Löwe S., Novikov E., Stahlbauer A., Wendler P. Precision Reuse for Efficient Regression Verification // ESEC/FSE 2013.

** Experimental branch with support of multi-property verification.

BenchExec Changes

Input

Verification task:

specification (M properties)

property_1.prp
…
property_M.prp

Error function unreachability

CHECK(init(main()), LTL(G ! call(error_1())))

…

Memory safety (composite property)

CHECK(init(main()), LTL(G valid-free))
CHECK(init(main()), LTL(G valid-deref))
CHECK(init(main()), LTL(G valid-memtrack))

checked program

source_1.c
…
source_R.c

Potential verifier options

-spec property_1.prp ...
-spec property_M.prp
source_1.c … source_R.c

Expected results:

YAML file

property_1: true|false
…
property_M: true|false

per each property
- determine category
- compute score

Output

Per each property

Status: true|false|unknown
Category: correct|wrong|...
Score: ...
Witnesses*: …
Termination reason*: …
Resource usage*: ...

Per verification task**

Status: true|false|unknown
Category: correct|wrong|...
Score: ...
Termination reason: …
Resource usage: ...

* May be useful in the future.
** In order to support backward compatibility.

Example of Resulting Table*

Status / category for task

Unknown for some property

Status / category for each property

Termination reasons

Expected: - true

Expected: - unknown

Correct results for all properties

ldv-multiproperty/	status (unreach-call_idr)	status (unreach-call_urb)	status (unreach-call_spinlock)	status (unreach-call_mutex)	status (unreach-call_sysfs)	status (unreach-call_module)	status (unreach-call_arch_io)	status	cputime (s)
68.c	true	true	unknown	true	true	true	false(unreach-call)	false(unreach-call)	1060
69.c	unknown	true	unknown	unknown	unknown	unknown	unknown	true	610
70.c	unknown	unknown	unknown	unknown	unknown	unknown	unknown	timeout	3680
71.c	false(unreach-call)	unknown	unknown	unknown	unknown	unknown	unknown	timeout	3600
72.c	true	true	unknown	true	false(unreach-call)	true	unknown	false(unreach-call)	1550
73.c	true	true	true	true	true	true	true	false(unreach-call)	1530
74.c	true	unknown	true	true	true	true	true	true	253
75.c	true	true	true	true	true	true	false(unreach-call)	false(unreach-call)	686
76.c	true	true	false(unreach-call)	true	true	true	unknown	false(unreach-call)	383
77.c	true	true	true	true	true	true	unknown	true	237
78.c	unknown	true	true	true	unknown	unknown	unknown	true	2730
79.c	true	unknown	true	true	true	true	true	true	202
80.c	true	false(unreach-call)	true	true	true	true	true	false(unreach-call)	255
81.c	true	false(unreach-call)	true	true	true	true	true	false(unreach-call)	103
82.c	true	false(unreach-call)	true	true	true	true	true	false(unreach-call)	1150
83.c	true	false(unreach-call)	true	true	true	true	true	false(unreach-call)	256
84.c	true	true	true	true	unknown	true	true	true	1380
85.c	true	true	true	true	true	unknown	true	false(unreach-call)	743
86.c	true	true	true	unknown	true	true	unknown	true	737
87.c	unknown	unknown	unknown	unknown	unknown	unknown	unknown	error (143)	3300
88.c	true	true	true	true	true	true	unknown	true	231
89.c	true	true	true	unknown	true	true	true	true	584
90.c	unknown	unknown	unknown	false(unreach-call)	unknown	unknown	unknown	timeout	3600
91.c	true	true	true	true	true	true	true	true	293
92.c	true	true	true	true	true	true	true	true	1140
93.c	true	true	false(unreach-call)	false(unreach-call)	true	true	true	false(unreach-call)	223
94.c	true	true	true	true	true	false(unreach-call)	true	false(unreach-call)	1340
95.c	true	true	true	true	true	unknown	true	true	526
96.c	true	true	true	true	true	true	true	true	338
97.c	true	true	true	false(unreach-call)	true	unknown	true	false(unreach-call)	240
98.c	true	true	true	true	true	true	true	true	854
99.c	true	true	true	unknown	true	true	true	true	392
ldv-multiproperty/	status (unreach-call_idr)	status (unreach-call_urb)	status (unreach-call_spinlock)	status (unreach-call_mutex)	status (unreach-call_sysfs)	status (unreach-call_module)	status (unreach-call_arch_io)	status	cputime (s)
total	99	99	99	99	99	99	99	99	121000
local summary	–	–	–	–	–	–	–	–	121000
correct results	80	78	75	75	77	75	70	29	19700
correct true	76	70	72	68	76	72	59	10	6190
correct false	4	8	3	7	1	3	11	19	13500
incorrect results	–	–	–	–	–	–	–	0	–
incorrect true	–	–	–	–	–	–	–	0	–
incorrect false	–	–	–	–	–	–	–	0	–
score (99 tasks, max score: 5506)	**156**	**148**	**147**	**143**	**153**	**147**	**129**	**4702**	–

Score for each property

Overall score

* Based on SV-COMP demo-category '*ldv-multiproperty*' (99 tasks, 30 properties).
Tasks were solved by means of CMAV (only 7 of 30 properties are shown).

Experiments Setup

- Tools
  - LDV Tools: branch *regression_verification**
  - BenchExec: branch *multiproperty***
  - CMAV: CPAchecker branch *cmav-2*, revision 24399
  - SDC: CPAchecker branch *muauto*, revision 20125
- Limitations
  - 900 CPU seconds (per 1 task and per 1 property)
  - 15GB of RAM and 13GB Java heap (per 1 verifier run)
- Tasks
  - 4041 Linux kernel modules (4.0-rc1)
  - 4100 Linux kernel modules (4.1-rc1)
  - 30 LDV properties (error function unreachability)

* Git repository: https://forge.ispras.ru/git/klever.git.

** Git repository: https://github.com/vmordan/benchexec.git.

Evaluation* (RMPV-1)

Method	Results			CPU time / speedup		Average speedup**
	Safe	Unsafe	Difference (%)	Solving tasks**	Overall (LDV Tools)	
Separated verification	118 703	667	–0 +0	3 889 000 1	6 742 000 1	1
MPV(CMAV)	117 162	634	**–1.36** +0.06	1 289 000 **3.02**	1 514 000 **4.45**	**7.86**
RMPV-1	116 980	618	**–1.5** +0.04	927 000 **4.2**	1 152 000 **5.85**	**20.2**
RMPV-1+MN	117 416	647	**–1.14** +0.06	794 000 **4.9**	1 019 000 **6.62**	**20.49**

Reuse DB size: 48Mb, max entry size: 212Kb

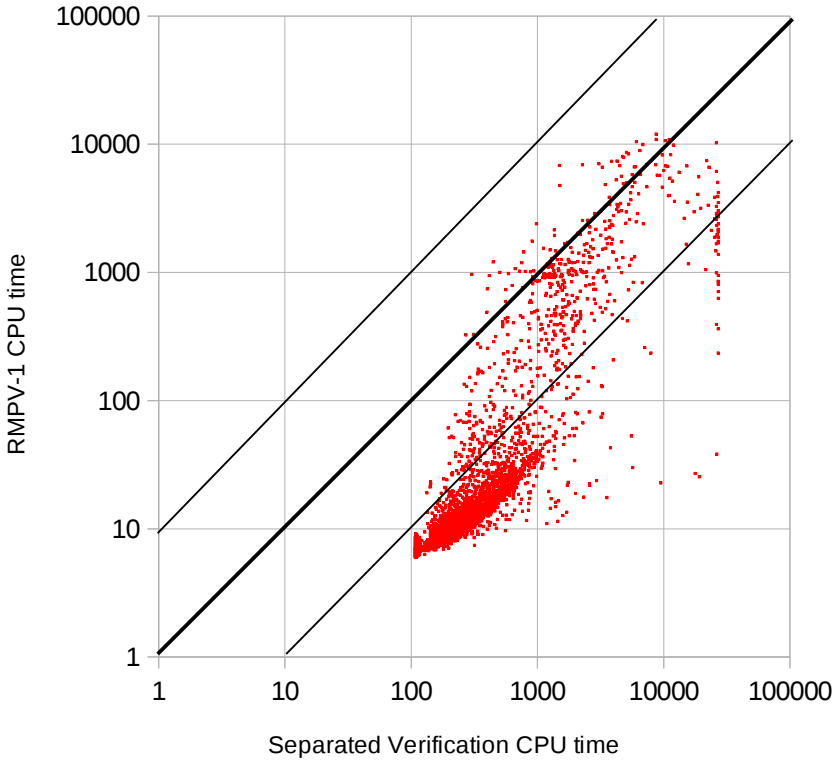
➡ Similar results with **1.62** efficiency increase

* Experiments were performed on the same version of Linux kernel modules (4.0-rc1).

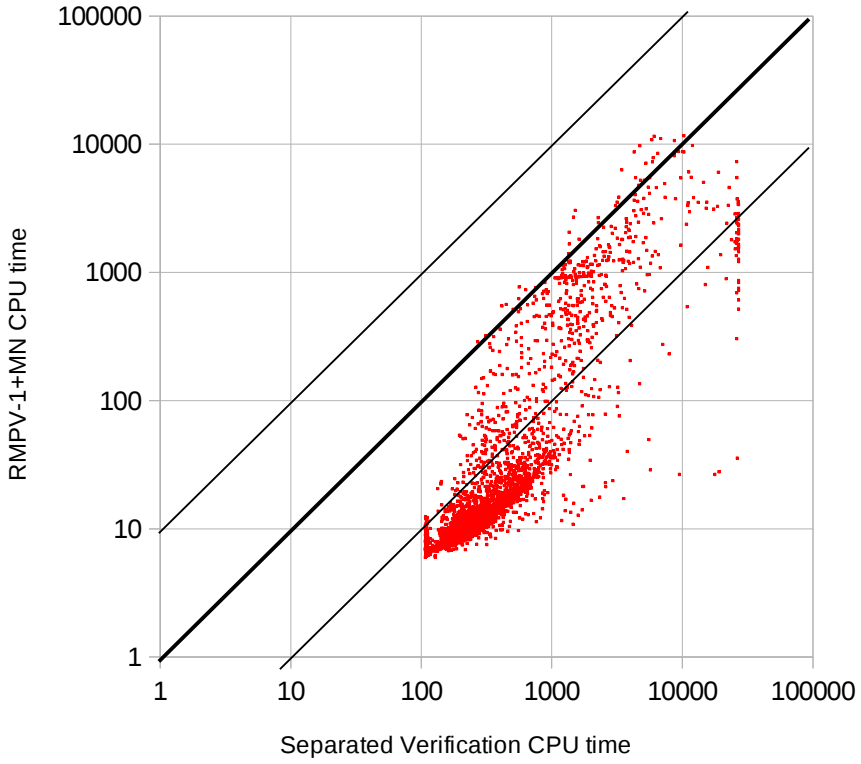
** These columns refer to CPAchecker CPU time (obtained by BenchExec).

Scatter Plots (RMPV-1)

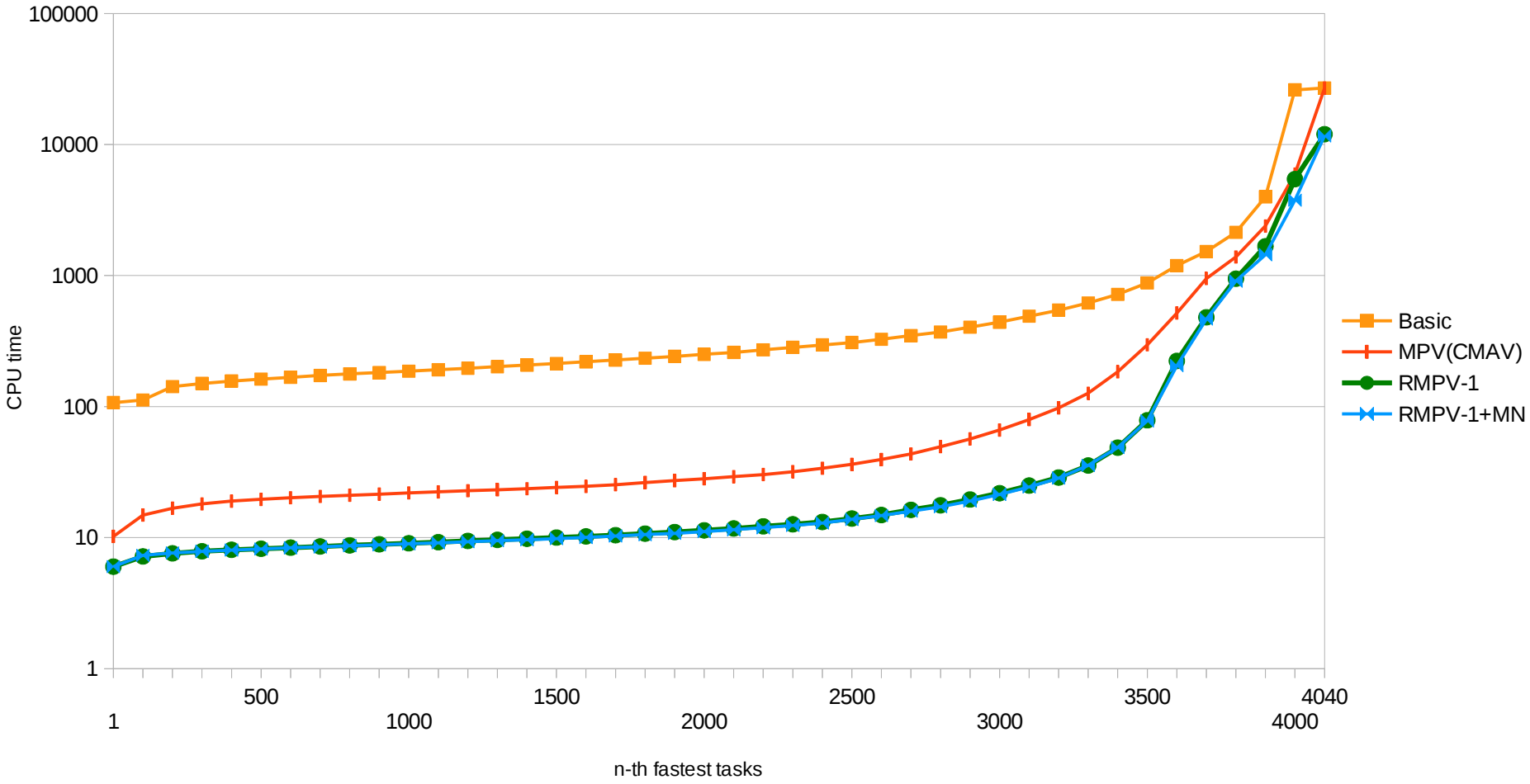
RMPV-1 vs. Basic



RMPV-1+MN vs. Basic



Quantile Plot (RMPV-1)



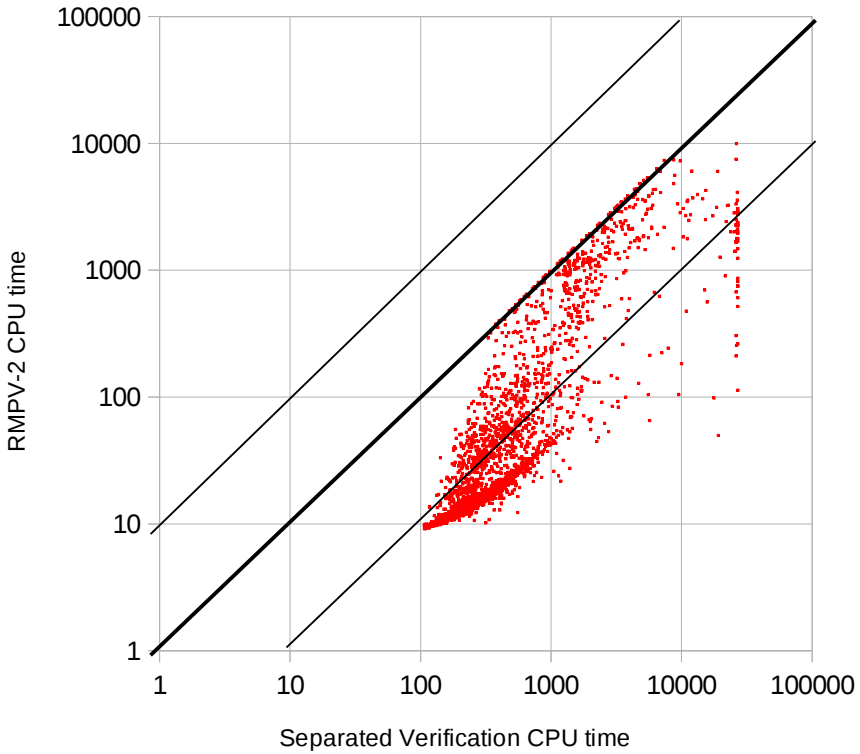
Evaluation (RMPV-2)

Method	Results			CPU time / speedup		Average speedup
	Safe	Unsafe	Difference (%)	Solving tasks	Overall	
Separated verification	118 703	667	–0 +0	3 889 000 1	6 742 000 1	1
MPV(CMAV)	117 162	634	**–1.36** +0.06	1 289 000 **3.02**	1 514 000 **4.45**	**7.86**
MPV(SDC)	118 386	673	**–0.45** +0.2	1 373 000 **2.83**	1 550 000 **4.35**	**13.42**
RMPV-2	118 149	675	**–0.68** +0.23	808 000 **4.81**	1 033 000 **6.53**	**14.06**
RMPV-1+MN	117 416	647	**–1.14** +0.06	794 000 **4.9**	1 019 000 **6.62**	**20.49**

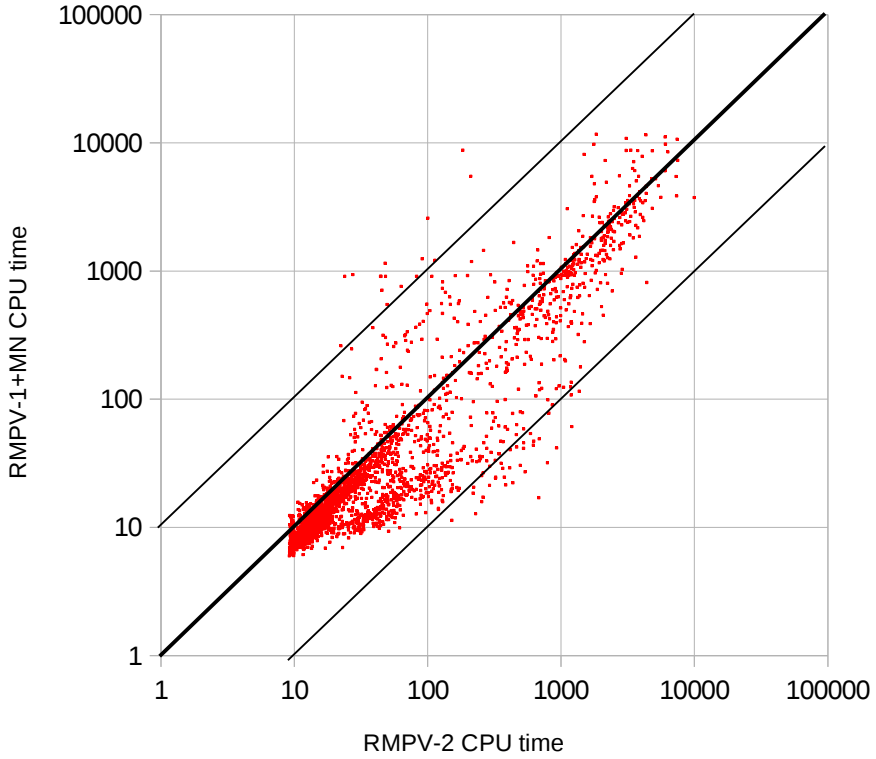
➡ RMPV-2: better results, RMPV-1: better speedup

Scatter Plots (RMPV-2)

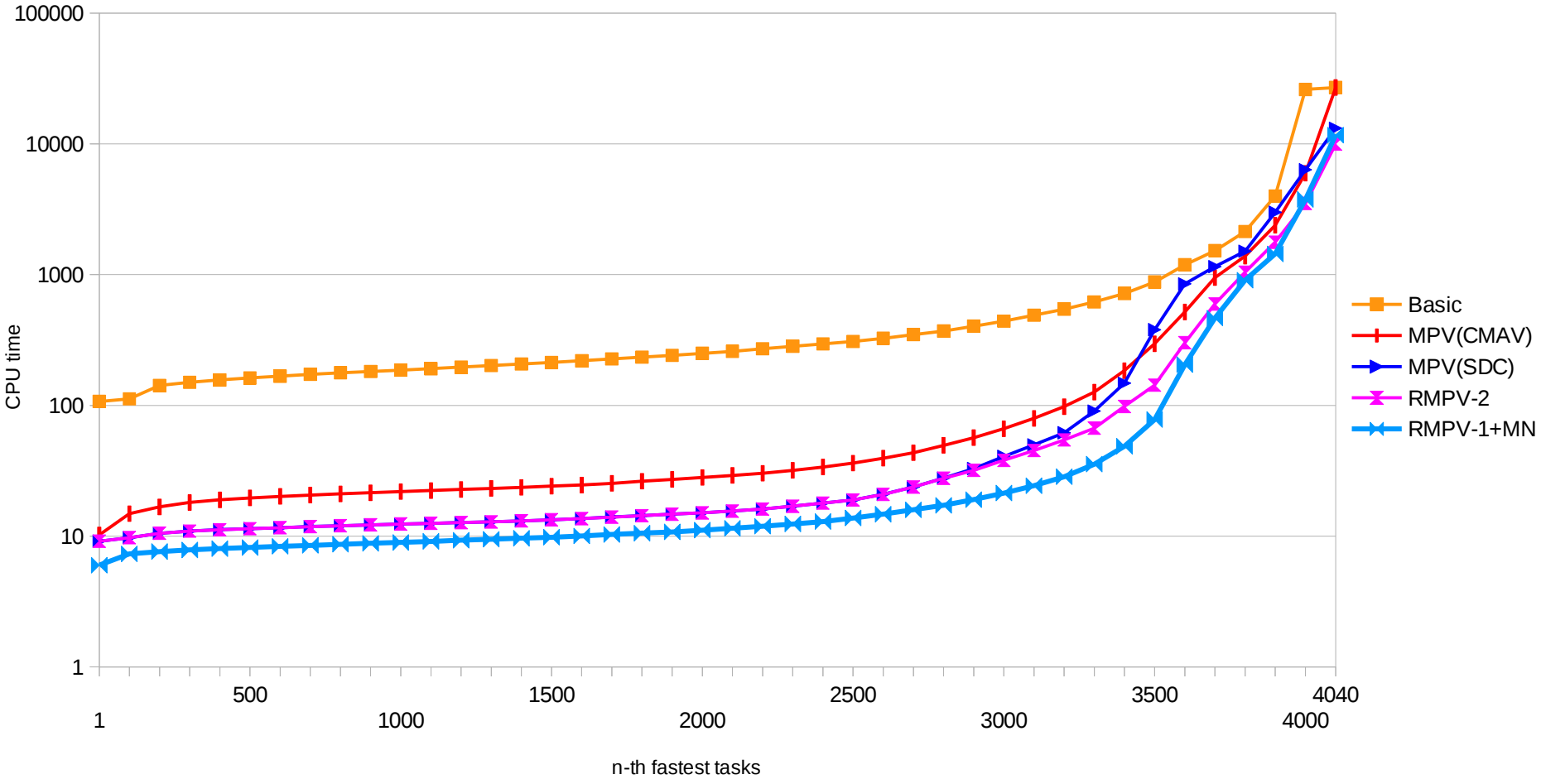
RMPV-2 vs. Basic



RMPV-1+MN vs. RMPV-2



Quantile Plot (RMPV-2)



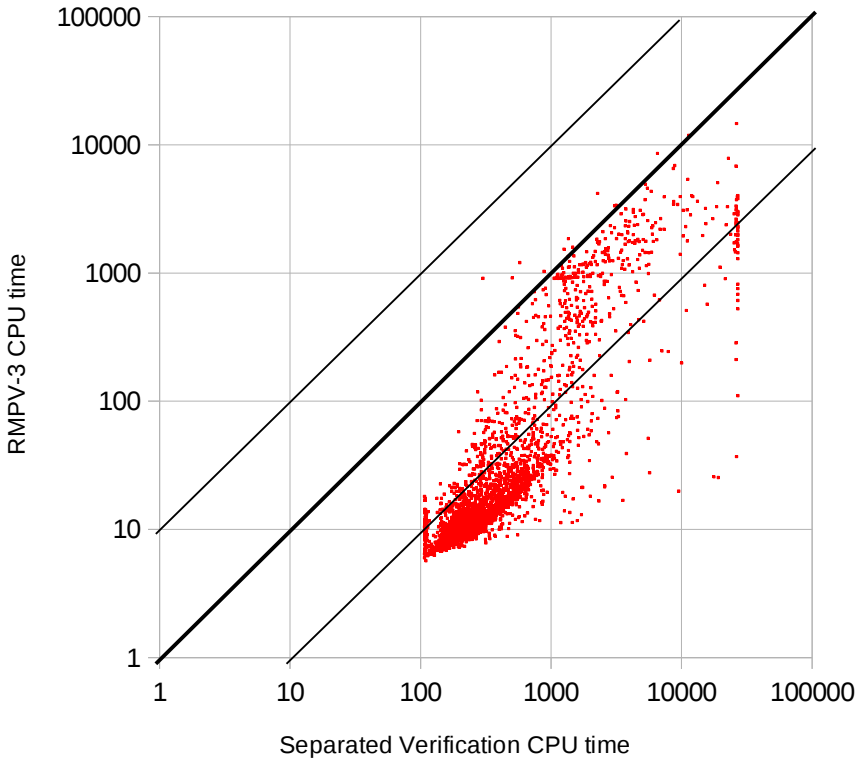
Evaluation (RMPV-3)

Method	Results			CPU time / speedup		Average speedup
	Safe	Unsafe	Difference (%)	Solving tasks	Overall	
Separated verification	118 703	667	–0 +0	3 889 000 1	6 742 000 1	1
MPV(CMAV)	117 162	634	**–1.36** +0.06	1 289 000 **3.02**	1 514 000 **4.45**	**7.86**
RMPV-1+MN	117 416	647	**–1.14** +0.06	794 000 **4.9**	1 019 000 **6.62**	**20.49**
RMPV-2	118 149	675	**–0.68** +0.23	808 000 **4.81**	1 033 000 **6.53**	**14.06**
RMPV-3	118 052	661	**–0.6** +0.06	658 000 **5.91**	883 000 **7.64**	**20.47**

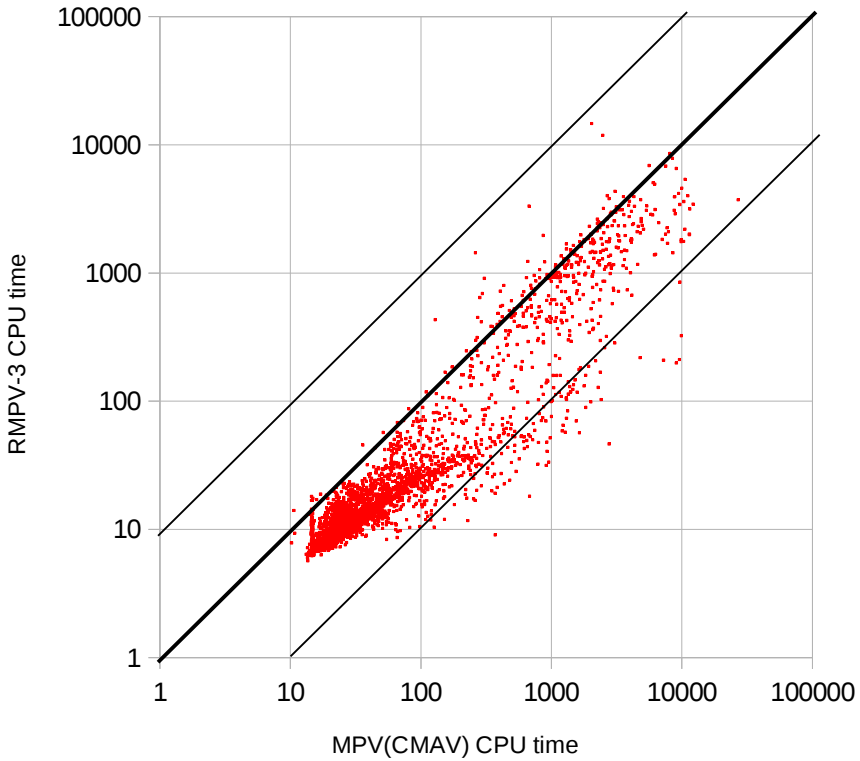
➡ Similar results with **2** efficiency increase

Scatter Plots (RMPV-3)

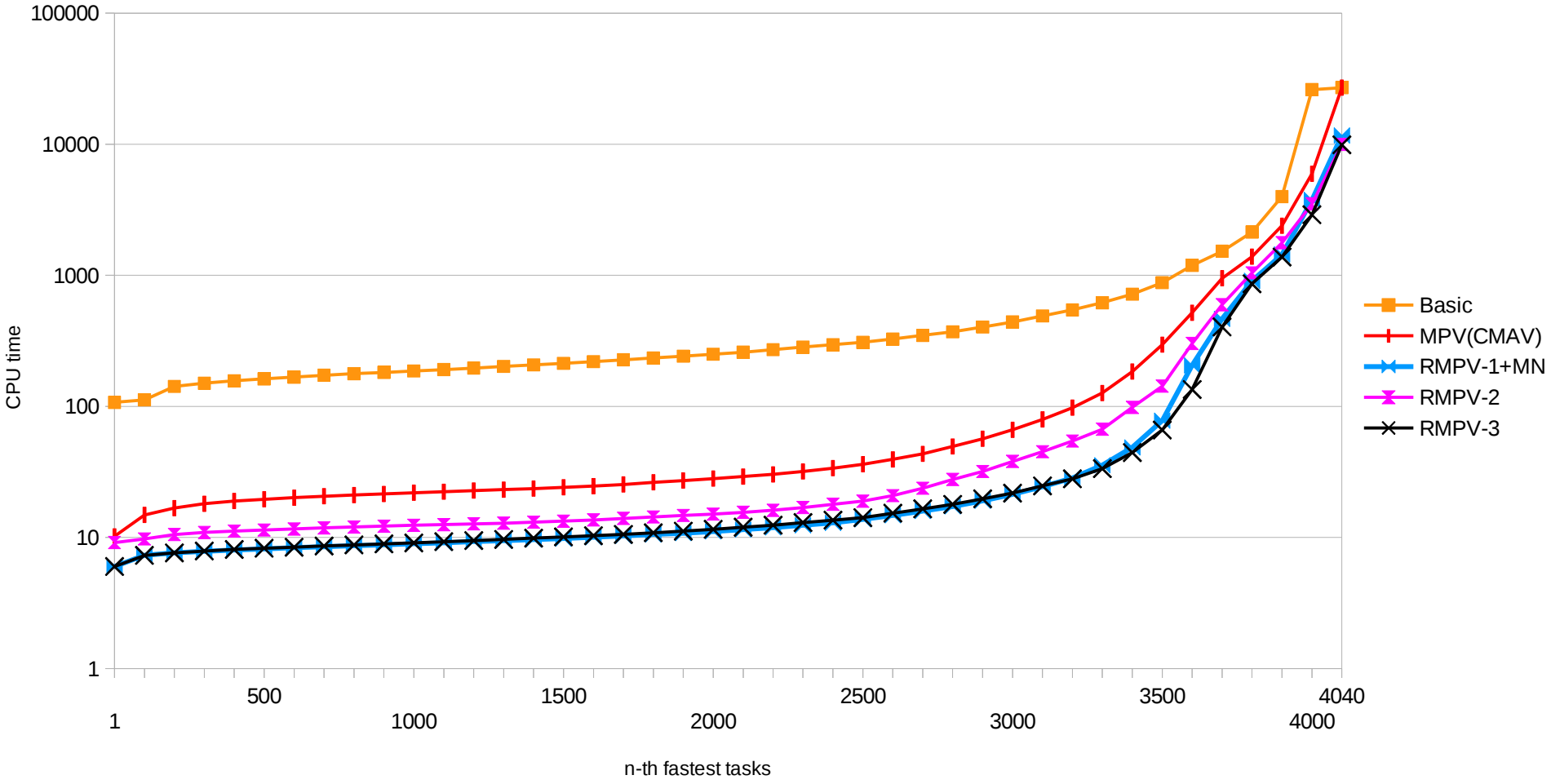
RMPV-3 vs. Basic



RMPV-3 vs. CMAV



Quantile Plot (RMPV-3)



Evaluation (Real-Life Conditions)

Method	Results			CPU time / speedup		Average speedup
	Safe	Unsafe	Difference (%)	Solving tasks	Overall	
Separated verification	120 493	676	–0 +0	4 067 000 1	7 120 000 1	1
MPV(CMAV)	118 716	643	**–1.55** +0.06	1 343 000 **3.03**	1 566 000 **4.55**	**7.87**
RMPV-3 (old)** *real-life conditions*	119 681	680	**–0.72** +0.05	779 000 **5.22**	1 017 000 **7**	**19.81**
RMPV-3 (updated***) *ideal conditions*	119 874	677	**–0.59** +0.08	662 000 **6.14**	907 000 **7.85**	**22.27**

Reuse DB size: 48Mb->49Mb, max entry size: 212Kb (same)

➡ RMPV-3 is applicable in real-life conditions

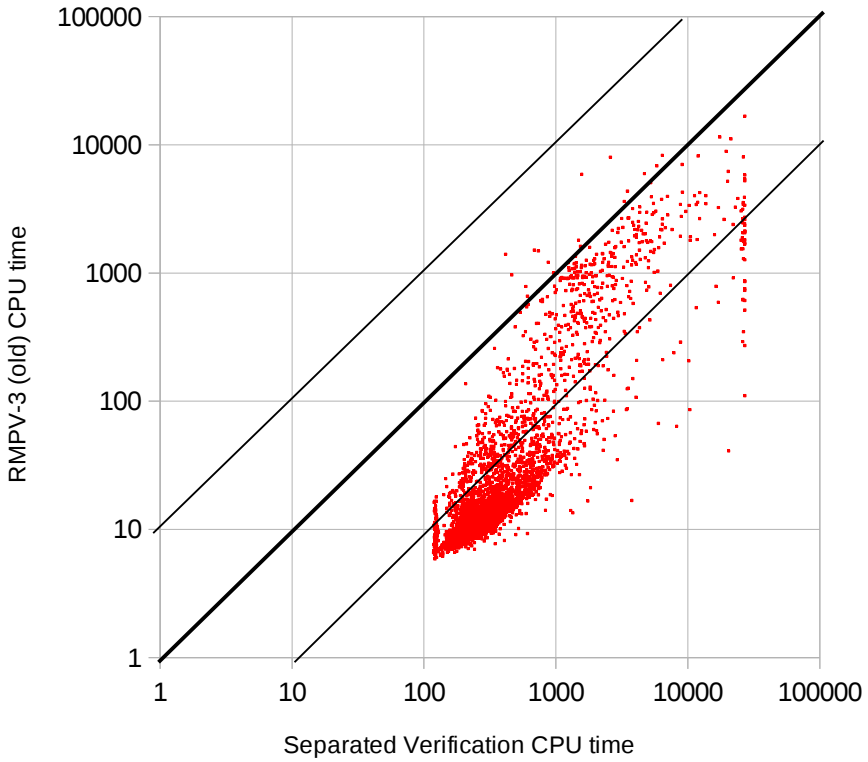
* For these experiments Linux kernel of version 4.1-rc1 was used.

** Verification of Linux kernel modules of version 4.1-rc1 based on precision, obtained on version 4.0-rc1.

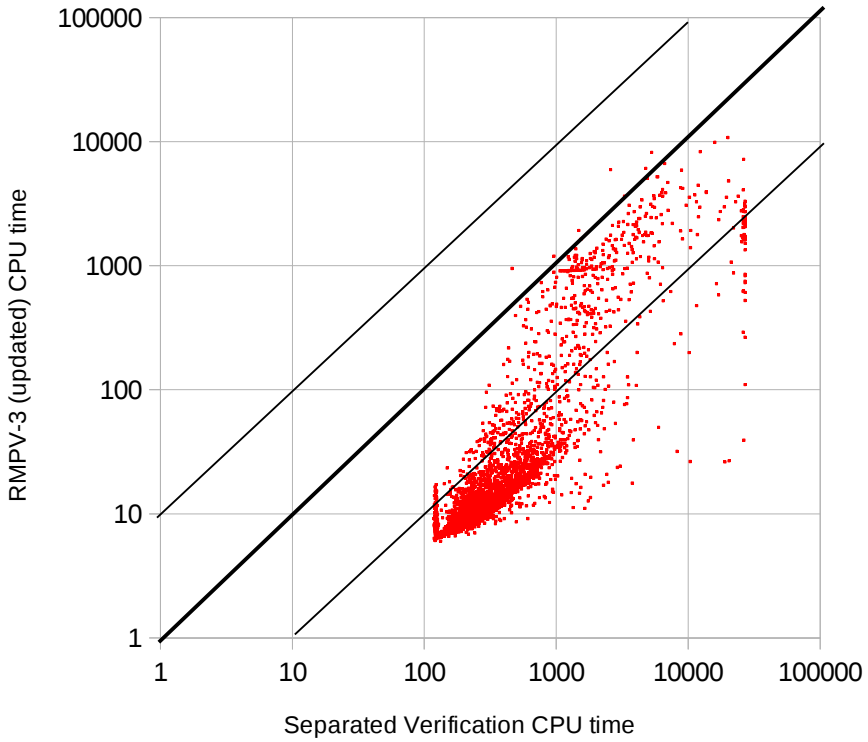
*** Verification of Linux kernel modules of version 4.1-rc1 based on precision, obtained on version 4.1-rc1.

Scatter Plots (Real-Life Conditions)

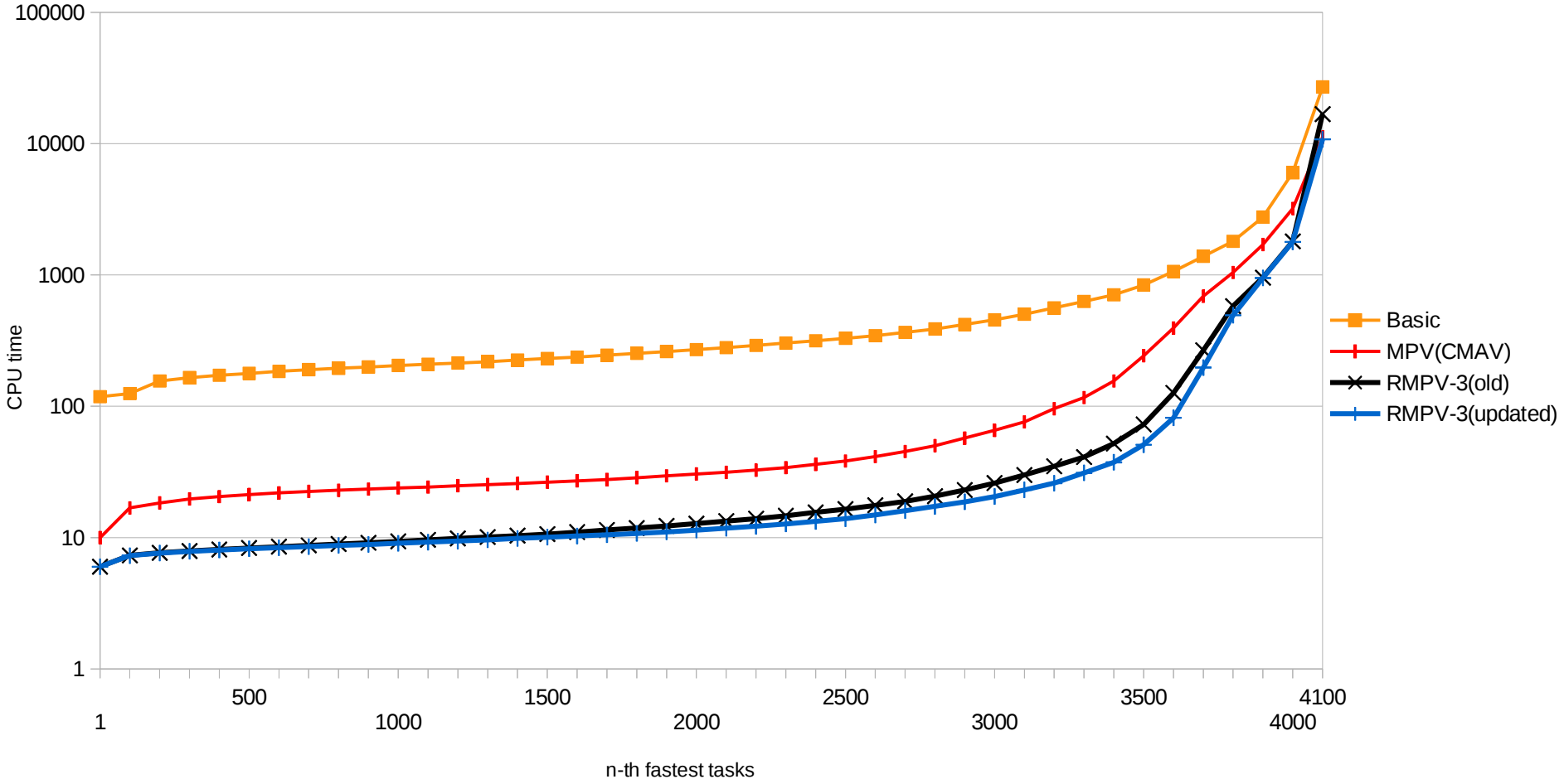
RMPV-3 (old) real-life conditions



RMPV-3 (updated) ideal conditions



Quantile Plot (Real-Life Conditions)



Conclusion

- Combination of software model checking methods
  - Aims at real software
  - Increases efficiency in several times
  - Requires minimum overheads

Future Plans

- New combinations of methods
  - Reuse other verification facts (witnesses, ARG, etc.)
  - Check other property kinds (termination, memory safety, etc.)
  - Focus on effectiveness over efficiency
  - Combine several verifiers (UAtomizer, SMACK, etc.)
- SV-COMP
  - Extend/improve demo-category 'ldv-multiproperty'

Thank you

Vitaly Mordan
mordan@ispras.ru

Institute for System Programming of the Russian Academy of Sciences