Performance-Detective: Automatic Deduction of Cheap and Accurate Performance Models

Larissa Schmid, Marcin Copik, Alexandru Calotoiu, Dominik Werle, Andreas Reiter, Michael Selzer, Anne Koziolek, Torsten Hoefler

Motivation: Cost of computing clusters

Brainware for green HPC

Christian Bischof · Dieter an Mey · Christian Iwainsky

Table 1 Tocal cost of ownership

Cost category	Cost/Year	Percentage
Building (7.5 Mio/25 years)	300,000 €	5.42%
Investment compute servers	2,000,000 €	36.14%
Hardware maintenance	800,000 €	14.46%
Power	1,563,660 €	28.26%
Linux	0 €	0.00%
Batch system	100,000 €	1.81%
ISV software	0 €	0.00%
HPC software	50,000 €	0.90%
Staff (12 FTE)	720,000 €	13.01%
Total sum	5,533,660 €	100.00%

Motivation: Cost of computing clusters

Karlsruhe Institute of Technology

Brainware for green HPC

Christian Bischof · Dieter an Mey · Christian Iwainsky

Table 1 Tocal cost of ownership

Cost category	Cost/Year	Percentage
Building (7.5 Mio/25 years)	300,000 €	5.42%
Investment compute servers	2,000,000 €	36.14%
Hardware maintenance	800,000 €	14.46%
Power	1,563,660 €	28.26%
Linux	0 €	0.00%
Batch system	100,000 €	1.81%
ISV software	0 €	0.00%
HPC software	50,000 €	0.90%
Staff (12 FTE)	720,000 €	13.01%
Total sum	5,533,660 €	100.00%

ExtraPeak: Advanced Automatic Performance Modeling for HPC Applications

Alexandru Calotoiu, Marcin Copik, Torsten Hoefler, Marcus Ritter, Sergei Shudler, and Felix Wolf

CPU hours that application optimization can achieve is enormous [5]. As the number of available cores increases at tremendous speed, reaping this potential is becoming an economic and scientific obligation. For example, an exascale system with a power consumption of 20 MW (very optimistic estimate) and 5000 h of operation per year would—assuming an energy price of 0.1€ per kWh—produce an energy bill of 10 M€ per year.

Ever-growing application complexity across all domains, including but not limited to theoretical physics, fluid dynamics, or climate research, requires a continuous

Motivation: Cost of computing clusters

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100

Building (7.5 Mio/25 years)	300,000 €	5.42%
Investment compute servers	2,000,000 €	36.14%
Hardware maintenance	800,000 €	14.46%
Power	1,563,660 €	28.26%
Linux	0 €	0.00%
Batch system	100,000 €	1.81%
ISV software	0 €	0.00%
HPC software	50,000 €	0.90%
Staff (12 FTE)	720,000 €	13.01%
Total sum	5,533,660 €	100.00%

CPU hours that application optimization can achieve is enormous [5]. As the number of available cores increases at tremendous speed, reaping this potential is becoming an economic and scientific obligation. For example, an exascale system with a power consumption of 20 MW (very optimistic estimate) and 5000 h of operation per year would—assuming an energy price of 0.1€ per kWh—produce an energy bill of 10 M€ per year.

Ever-growing application complexity across all domains, including but not limited to theoretical physics, fluid dynamics, or climate research, requires a continuous

Motivation: Configuration options

Performance-Influence Models for Highly Configurable Systems

Norbert Siegmund[†], Alexander Grebhahn[†], Sven Apel[†], Christian Kästner[‡]
[†]University of Passau, Germany [‡]Carnegie Mellon University, USA

Almost every complex software system today is configurable. While configurability has many benefits, it challenges performance prediction, optimization, and debugging. Often, the influences of individual configuration options on performance are unknown. Worse, configuration options may interact, giving rise to a configuration space of possibly exponential size. Addressing this challenge, we propose an approach that derives a *performance-influence model* for a

Motivation: Configuration options

Karlsruhe Institute of Technology

Performance-Influence Models for Highly Configurable Systems

Norbert Siegmund[†], Alexander Grebhahn[†], Sven Apel[†], Christian Kästner[‡]
[†]University of Passau, Germany [‡]Carnegie Mellon University, USA

Almost every complex software system today is configurable. While configurability has many benefits, it challenges performance prediction, optimization, and debugging. Often, the influences of individual configuration options on performance are unknown. Worse, configuration options may interact, giving rise to a configuration space of possibly exponential size. Addressing this challenge, we propose an approach that derives a *performance-influence model* for a

A Regression-Based Approach to Scalability Prediction*

Bradley J. Barnes
Dept. of Computer Science
The University of Georgia
barnes@cs.uga.edu

Barry Rountree
Dept. of Computer Science
The University of Georgia
rountree@cs.uga.edu

David K. Lowenthal
Dept. of Computer Science
The University of Georgia
dkl@cs.uga.edu

Jaxk Reeves
Department of Statistics
The University of Georgia
jaxk@stat.uga.edu

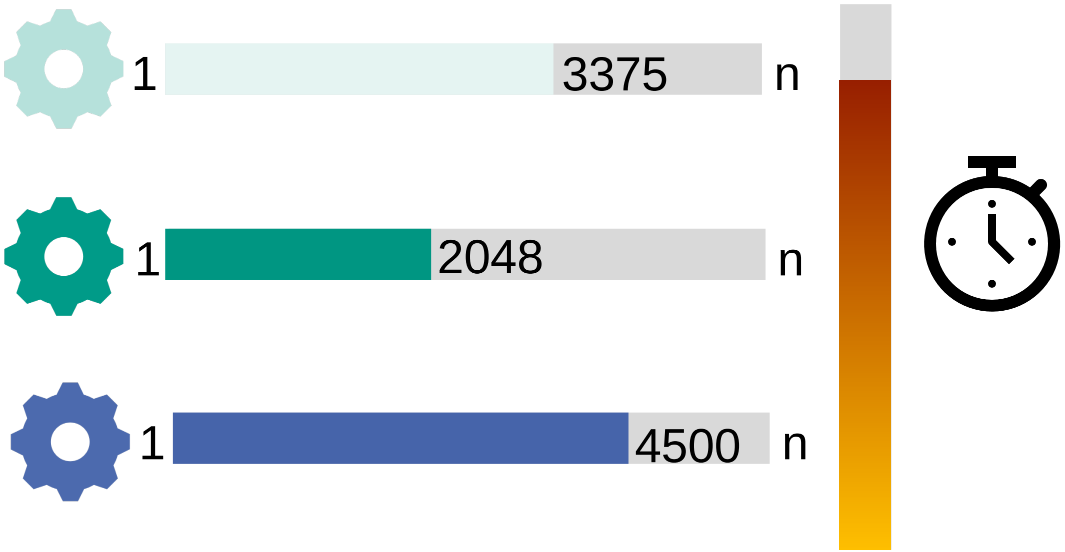
Bronis de Supinski
Lawrence Livermore
National Laboratory
bronis@llnl.gov

Martin Schulz
Lawrence Livermore
National Laboratory
schulzm@llnl.gov

Many applied scientific domains are increasingly relying on large-scale parallel computation. Consequently, many large clusters now have thousands of processors. However, the ideal number of processors to use for these scientific applications varies with both the input variables and the machine under consideration, and predicting this processor count is rarely straightforward. Accurate prediction mechanisms would provide many benefits, including improving cluster efficiency and identifying system configuration or

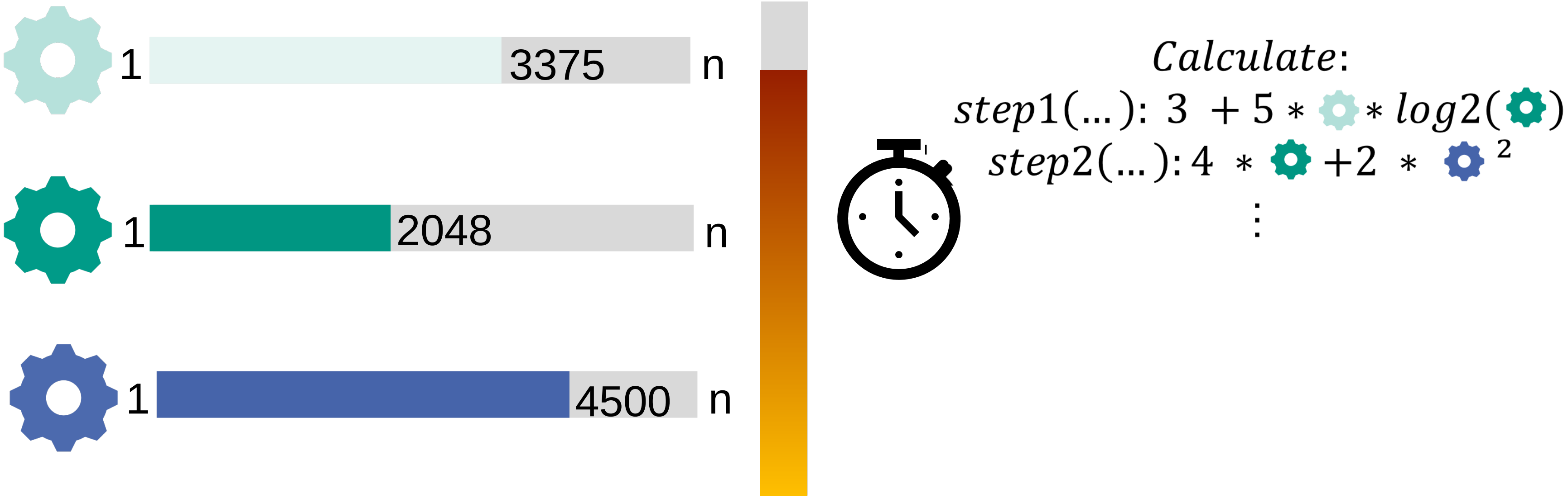
Performance Modeling

- Various options that influence performance
- How to choose the best configuration?



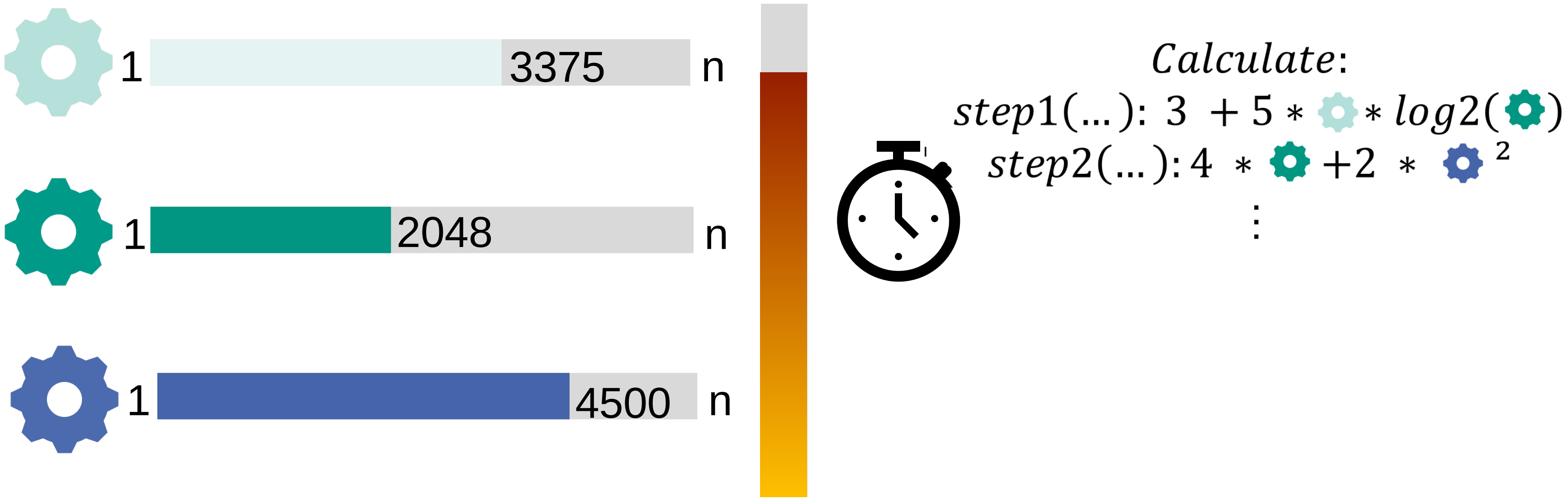
Performance Modeling

- Performance models help in understanding application behavior



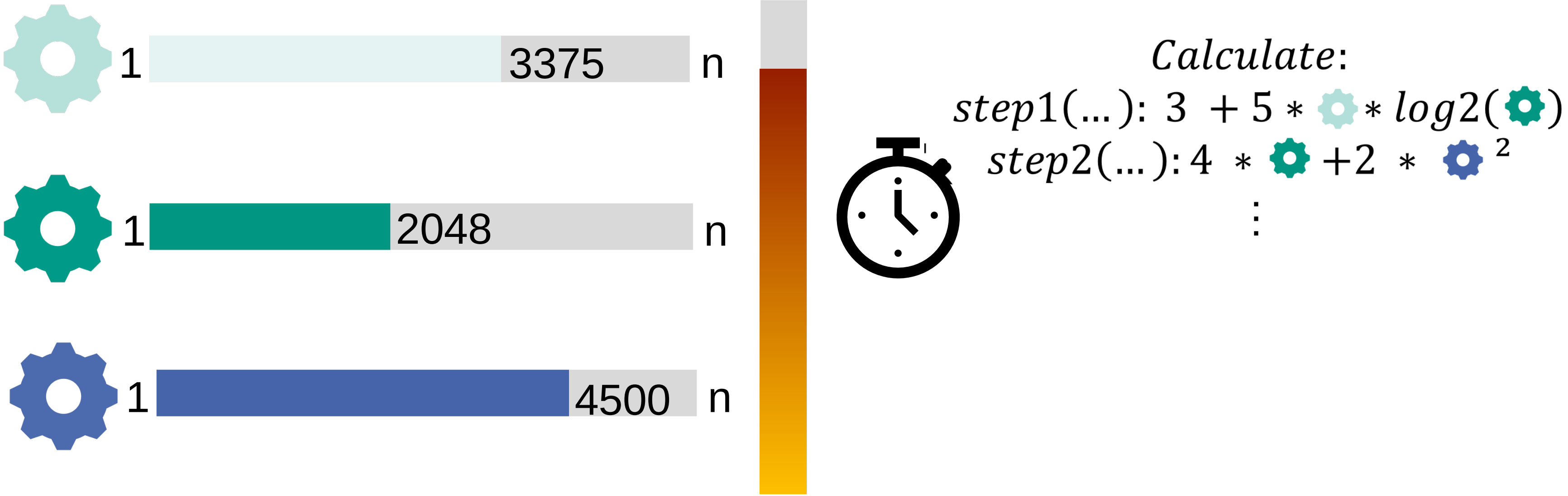
Performance Modeling

- Automatic performance modeling generates models from empirical measurements

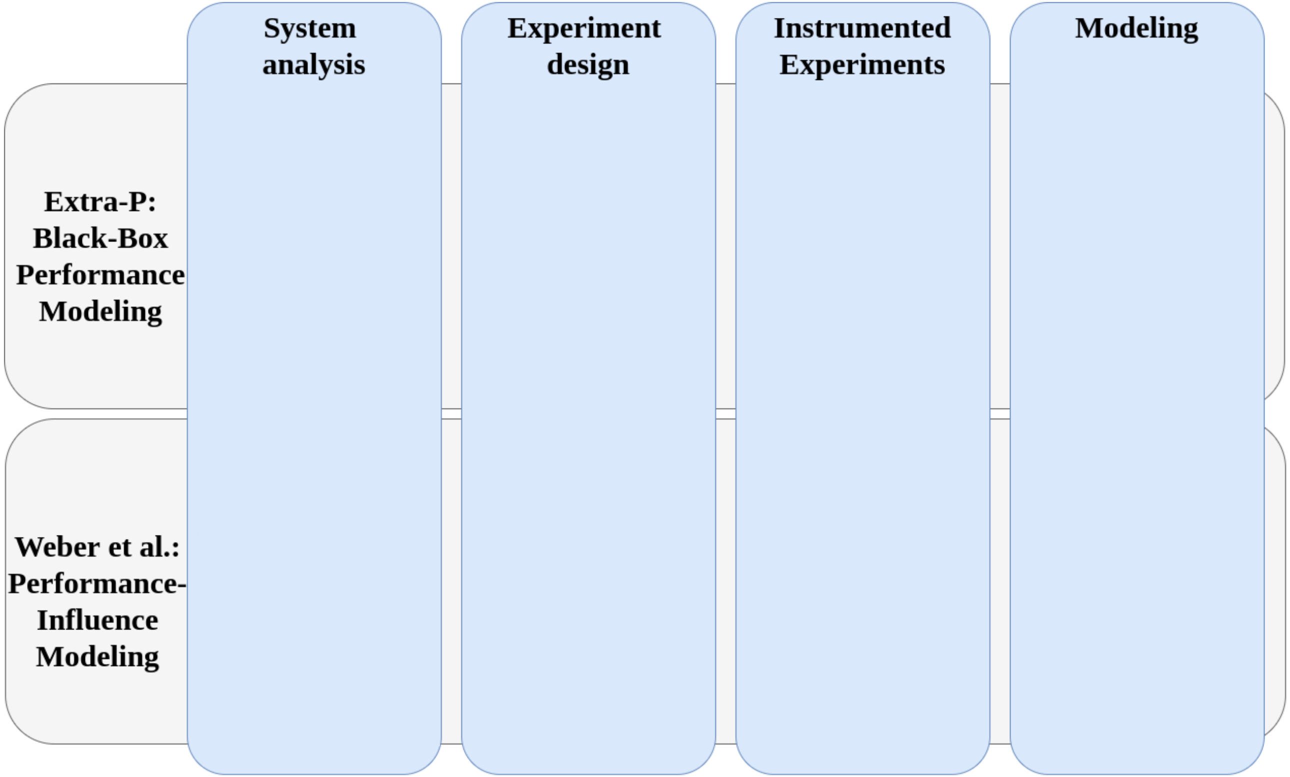


Performance Modeling

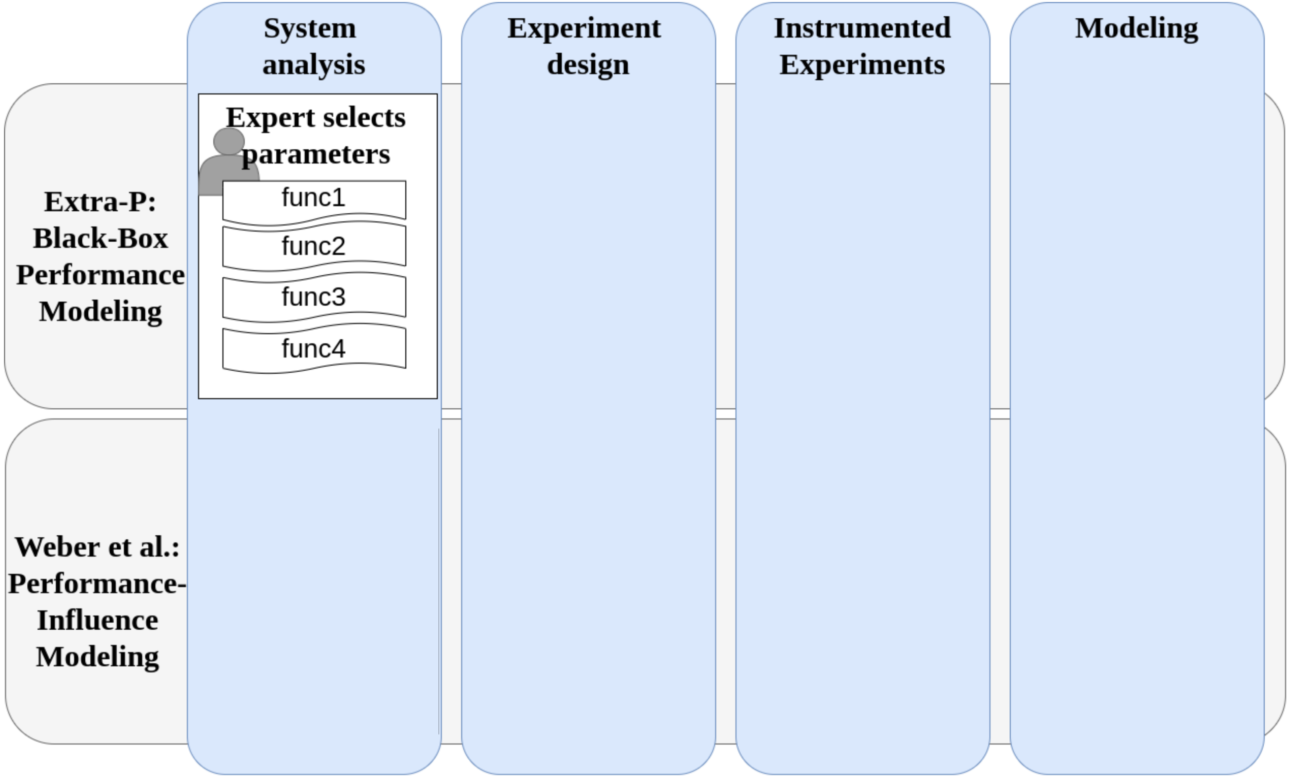
- Automatic performance modeling generates models from empirical measurements
- But – which strategy to use to select configurations to measure?



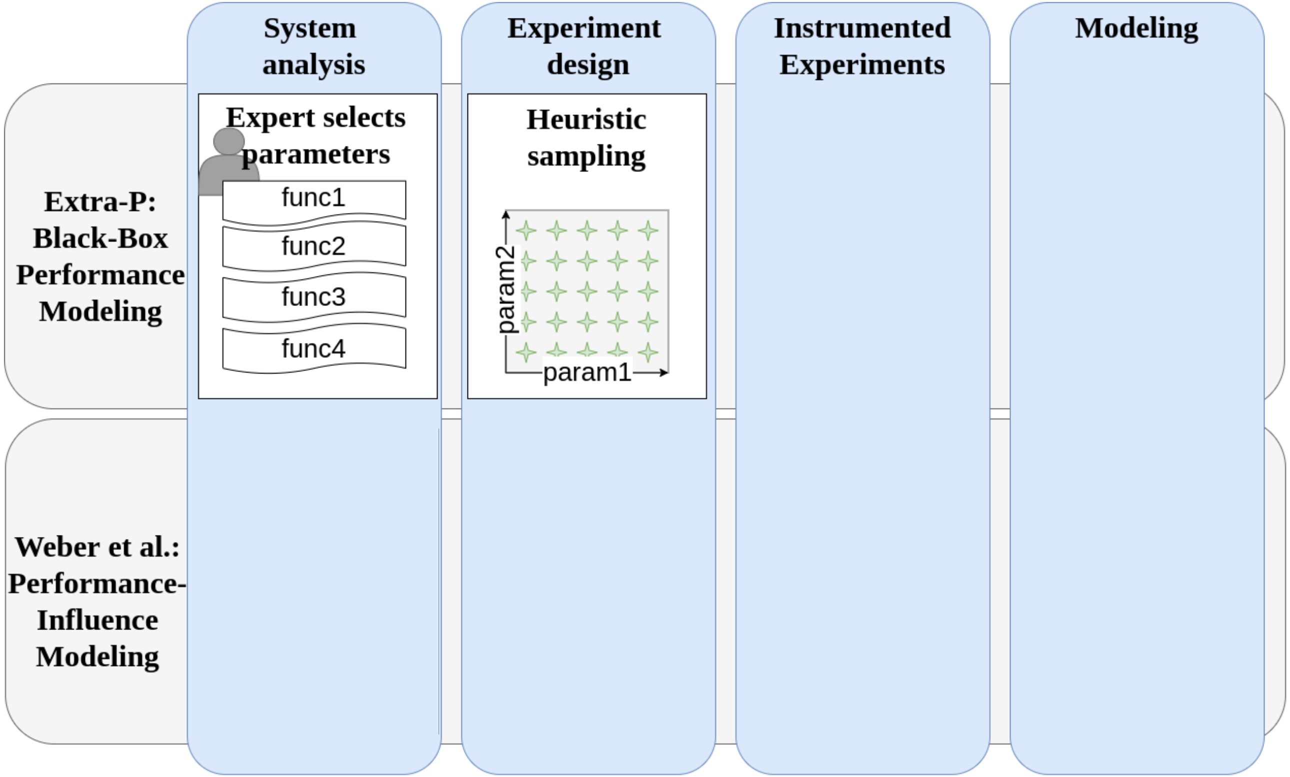
Current Modeling Workflows



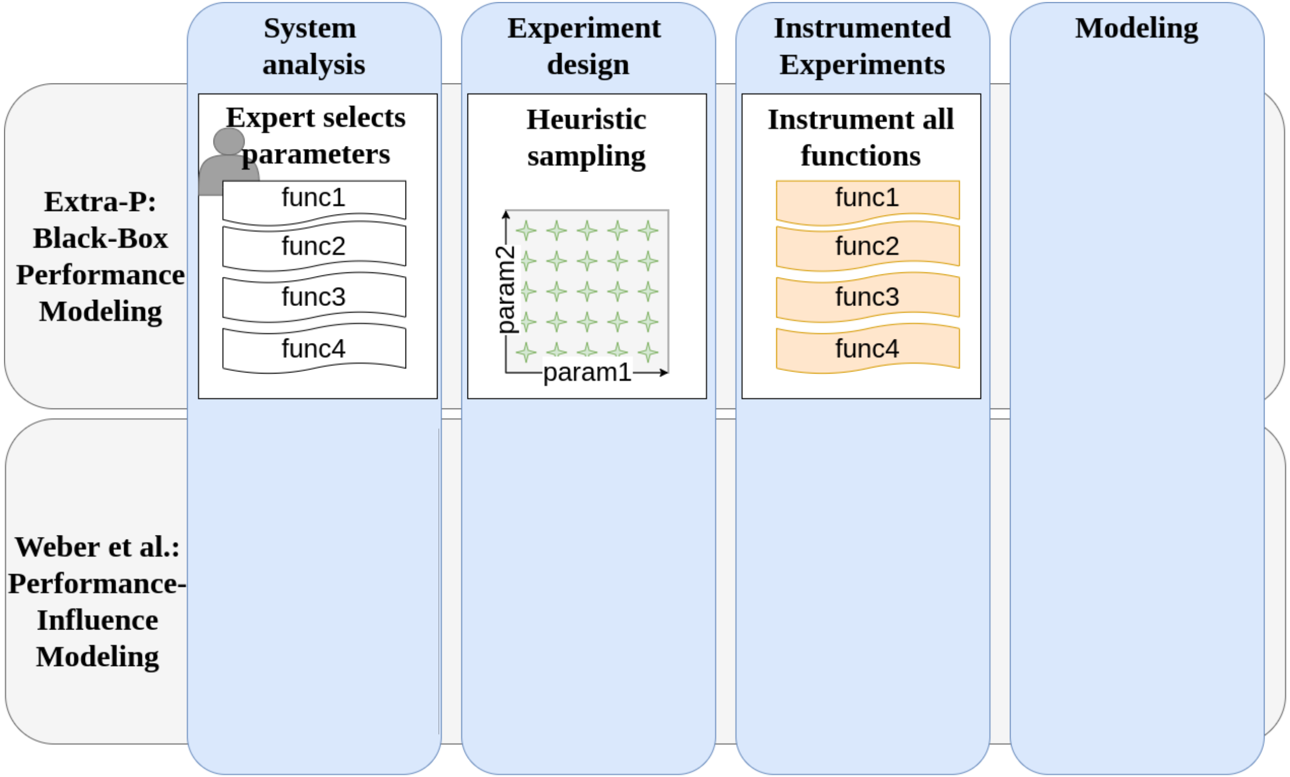
Current Modeling Workflows



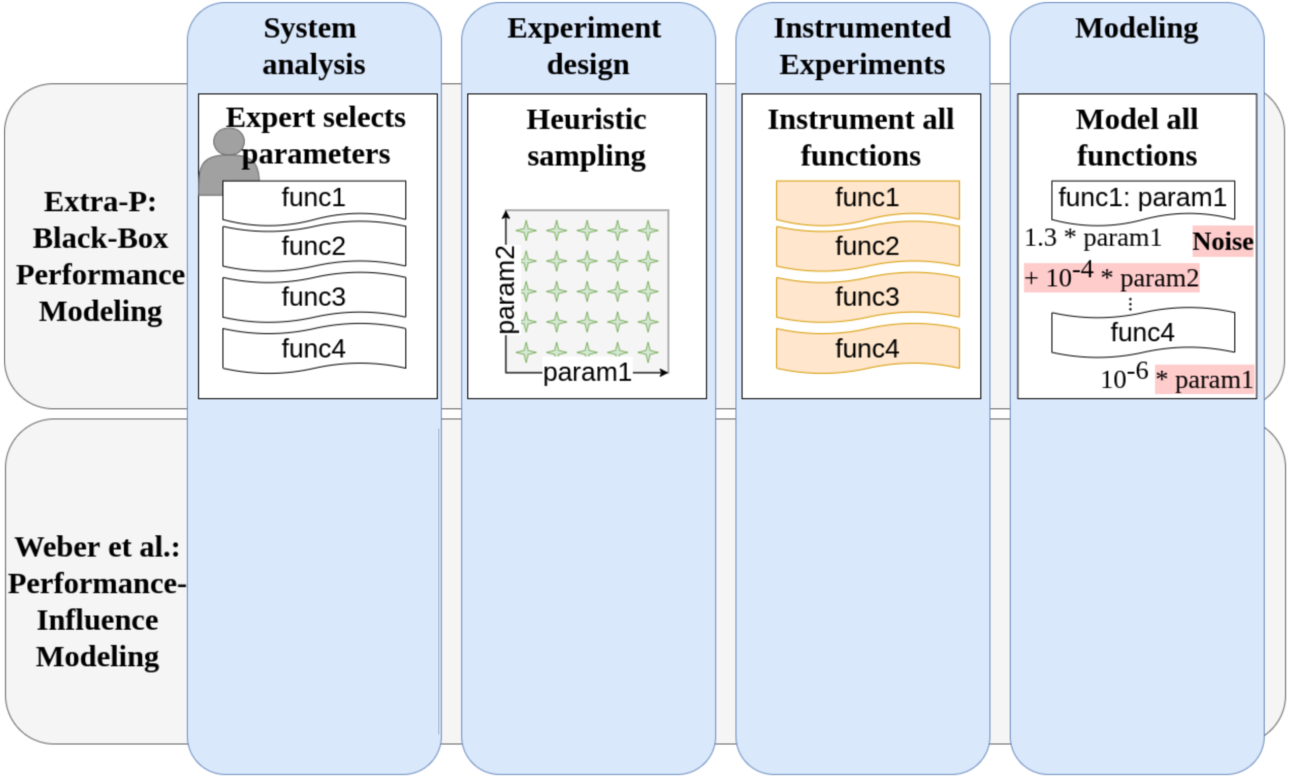
Current Modeling Workflows



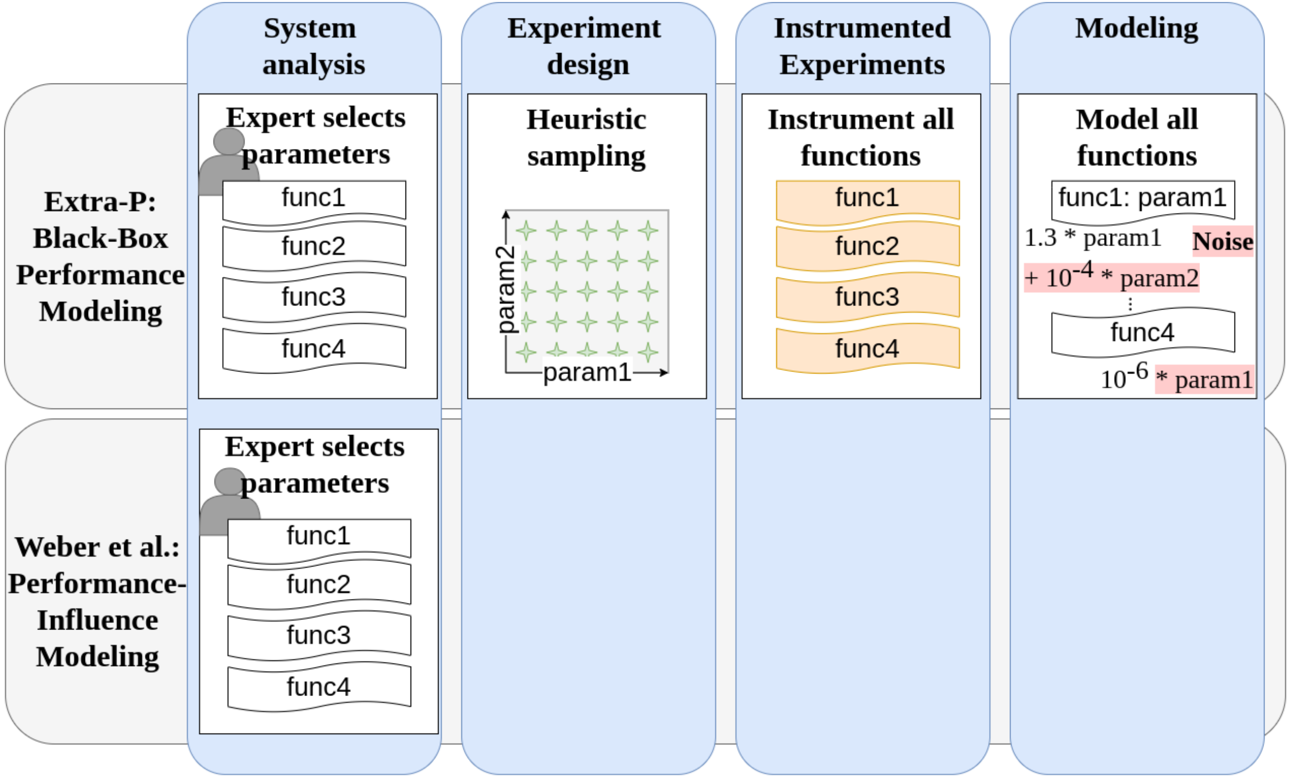
Current Modeling Workflows



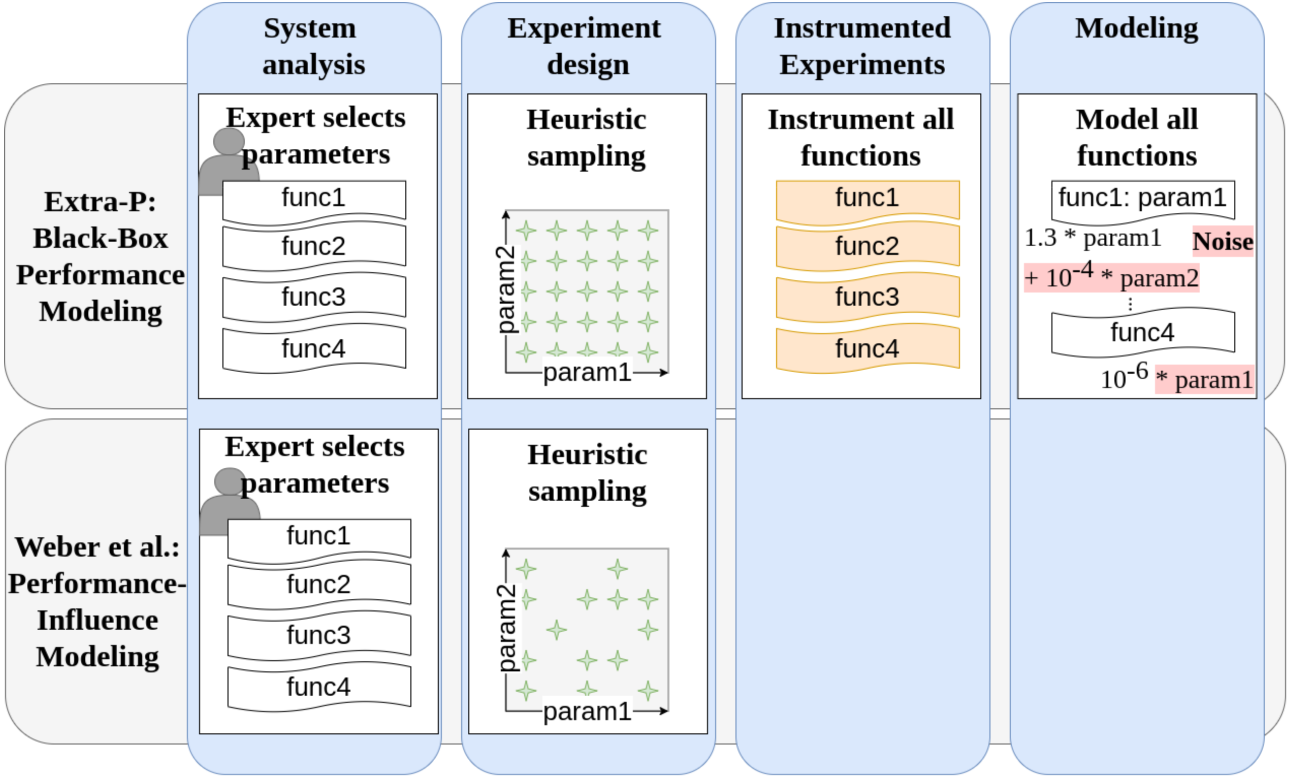
Current Modeling Workflows



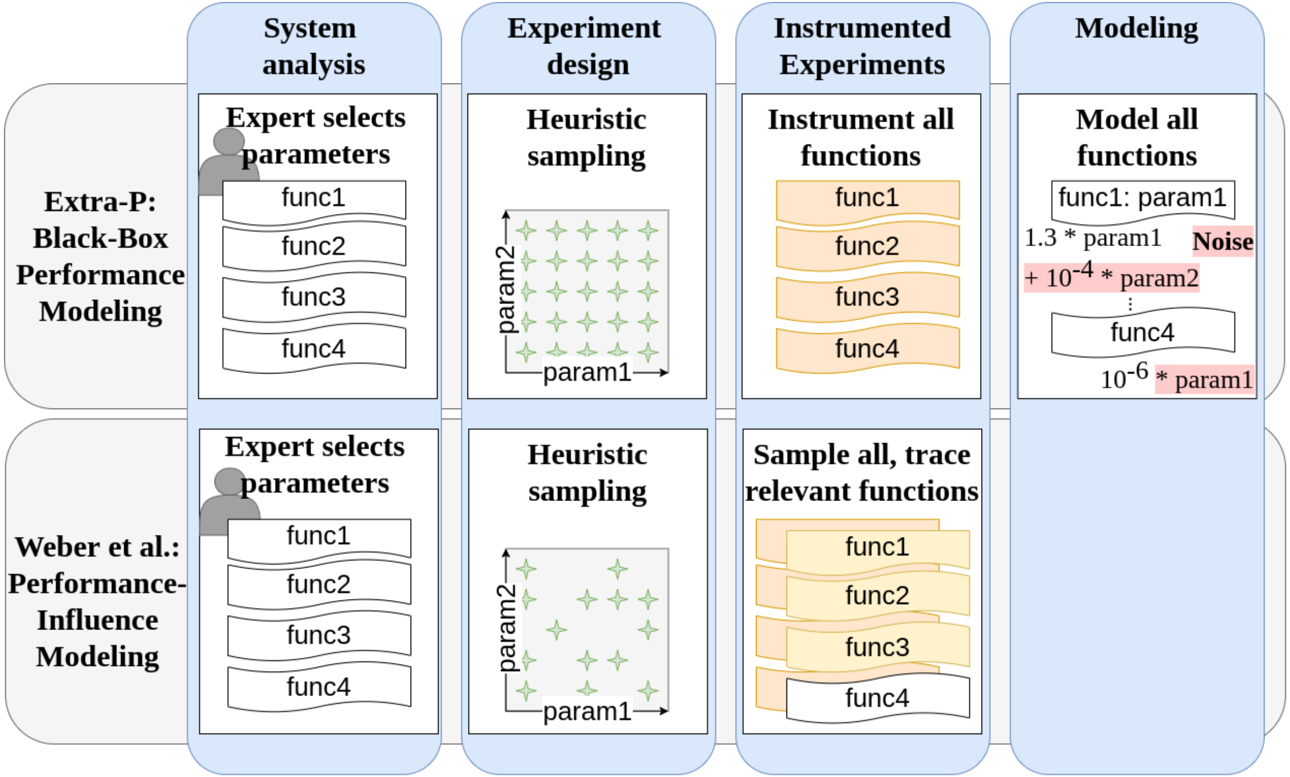
Current Modeling Workflows



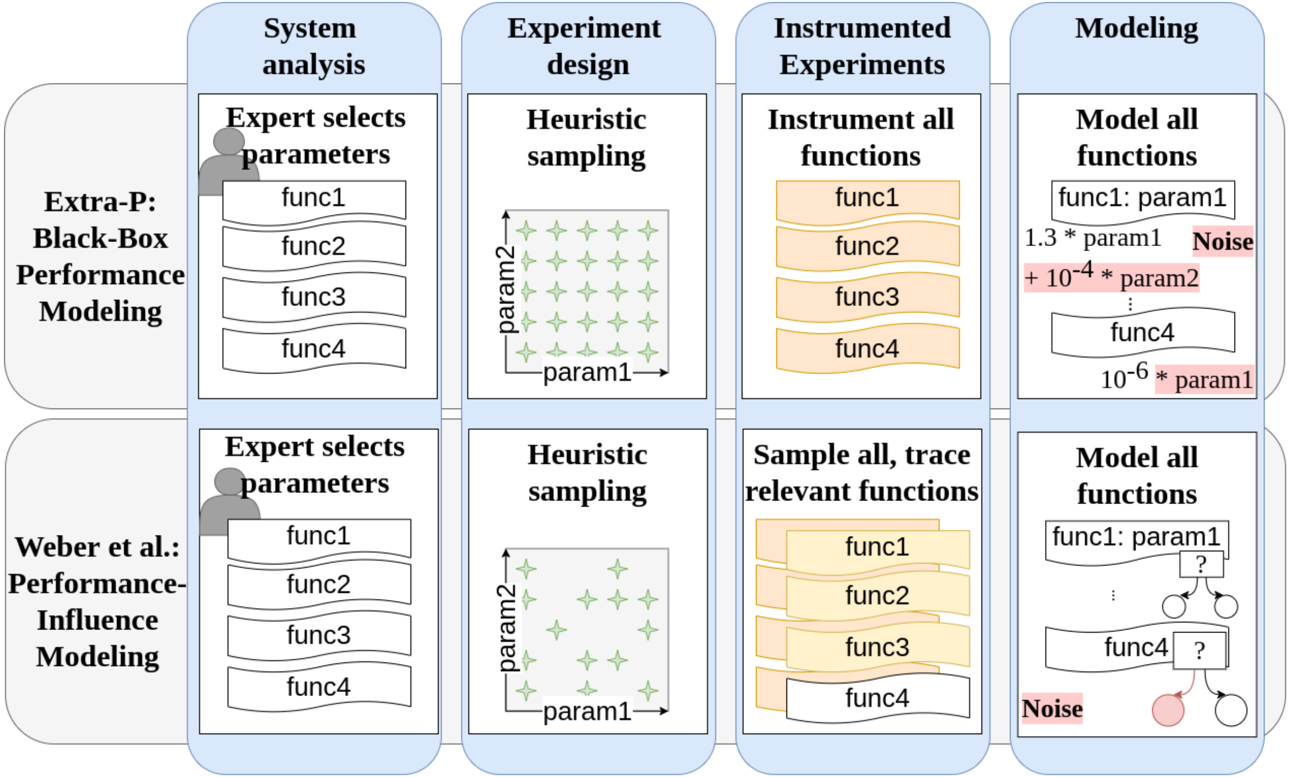
Current Modeling Workflows



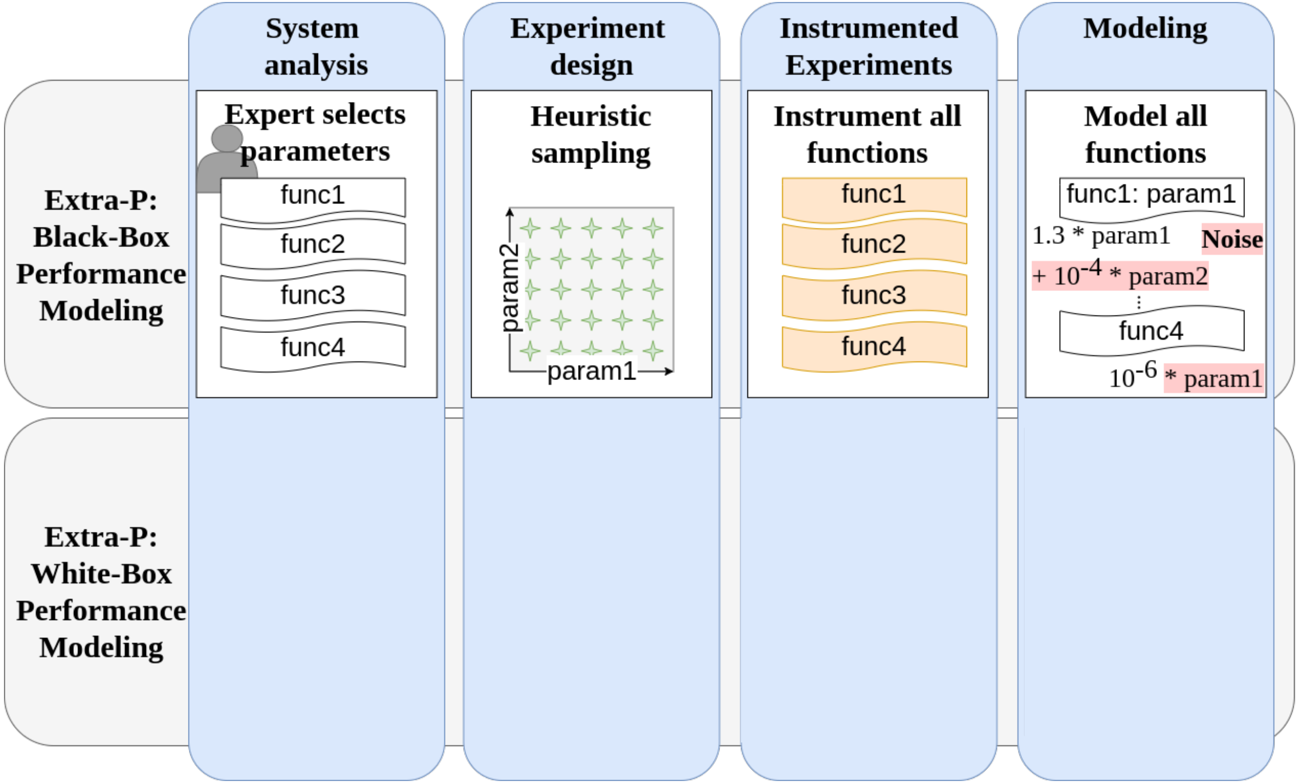
Current Modeling Workflows



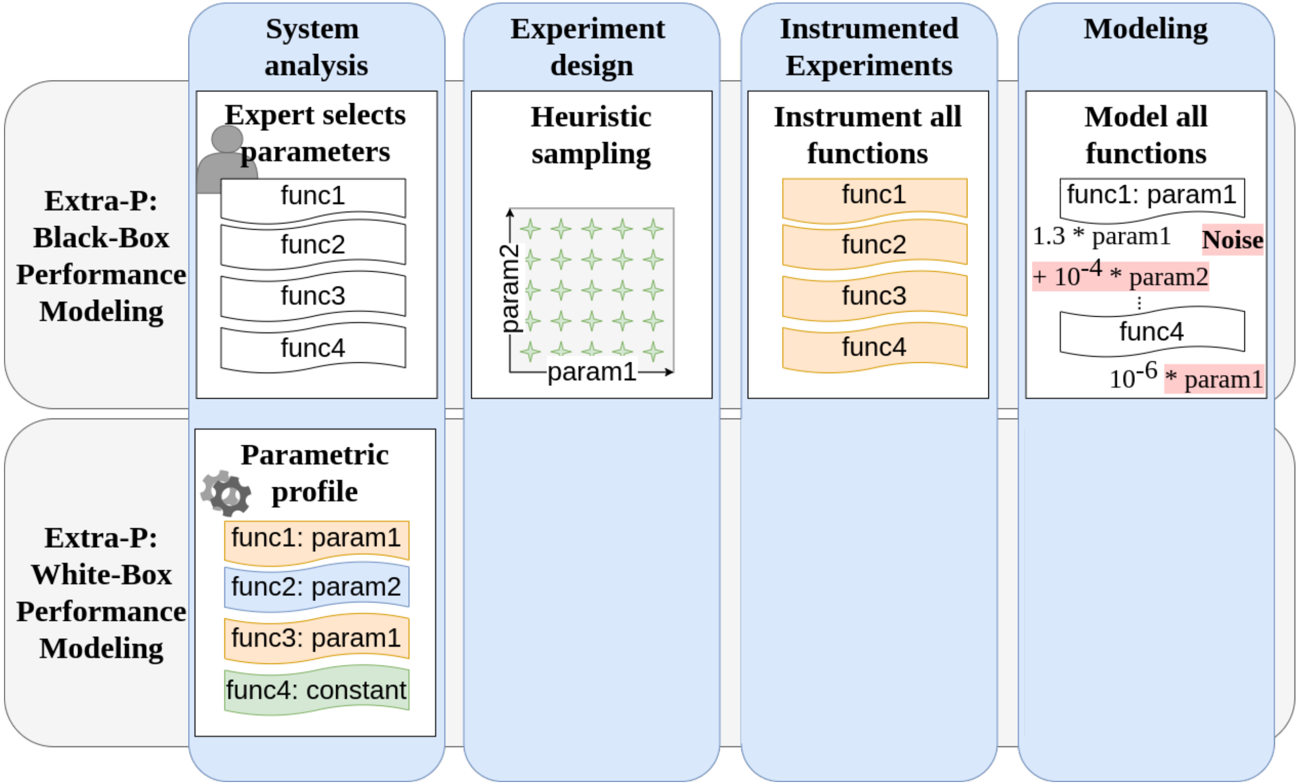
Current Modeling Workflows



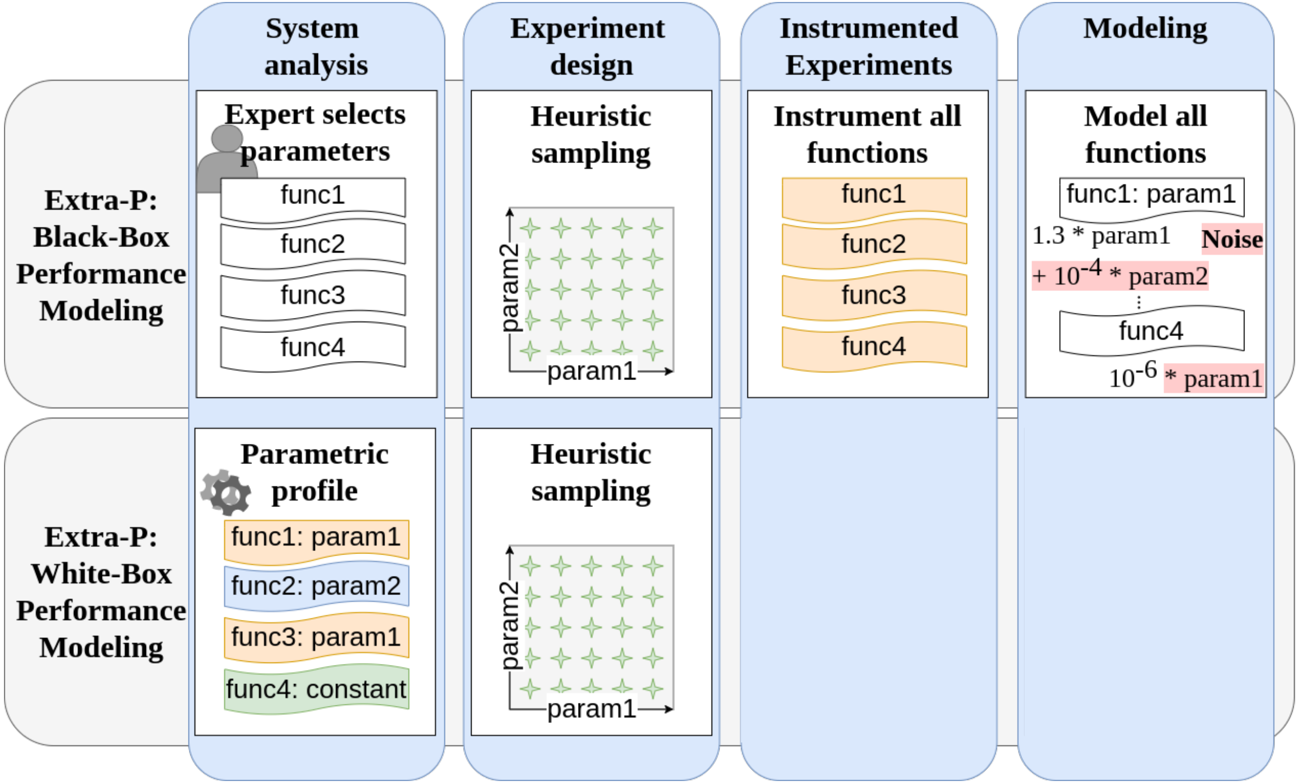
Current Modeling Workflows



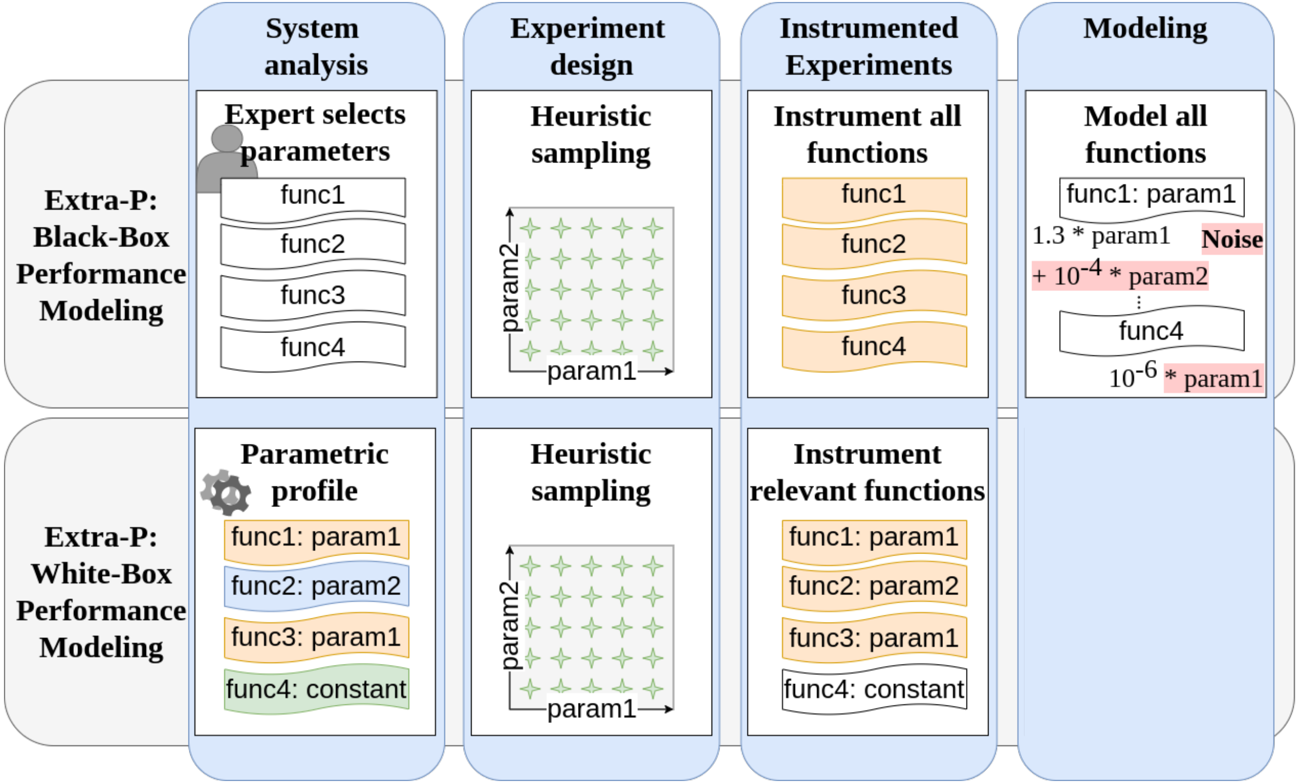
Current Modeling Workflows



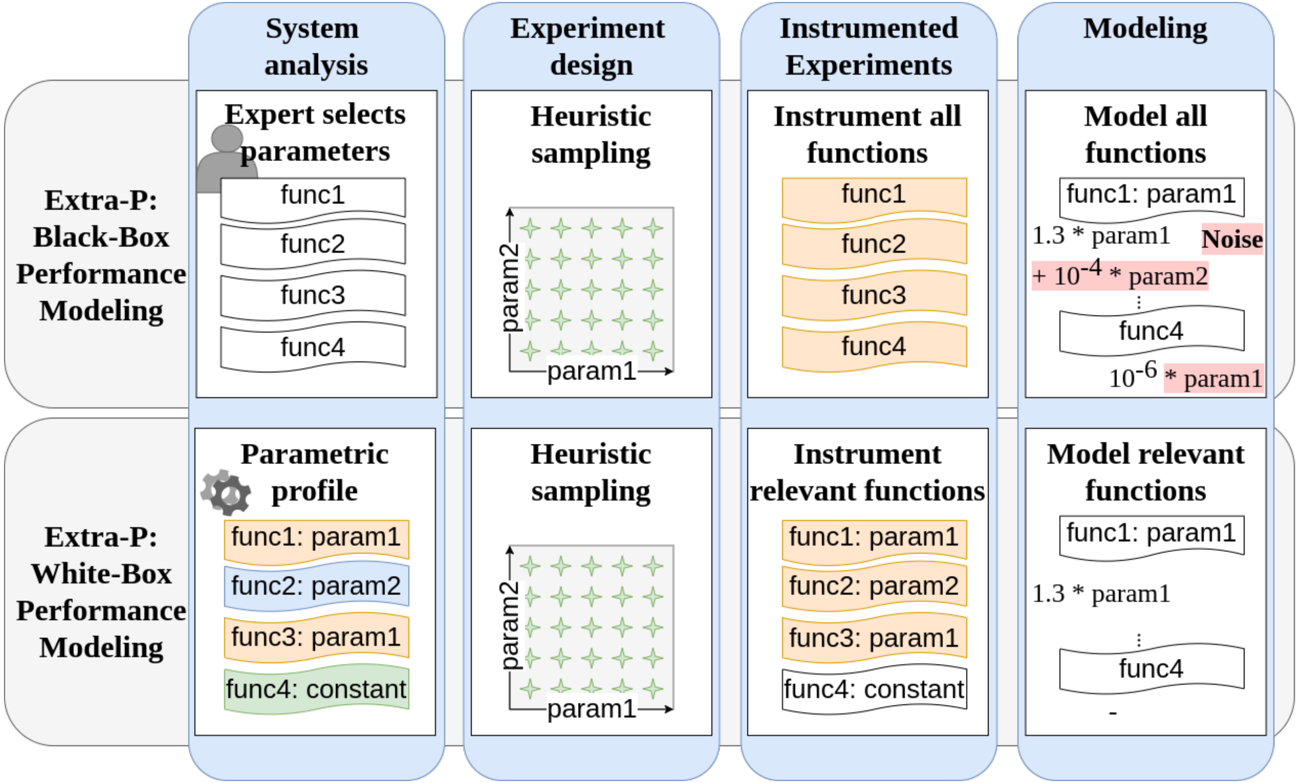
Current Modeling Workflows



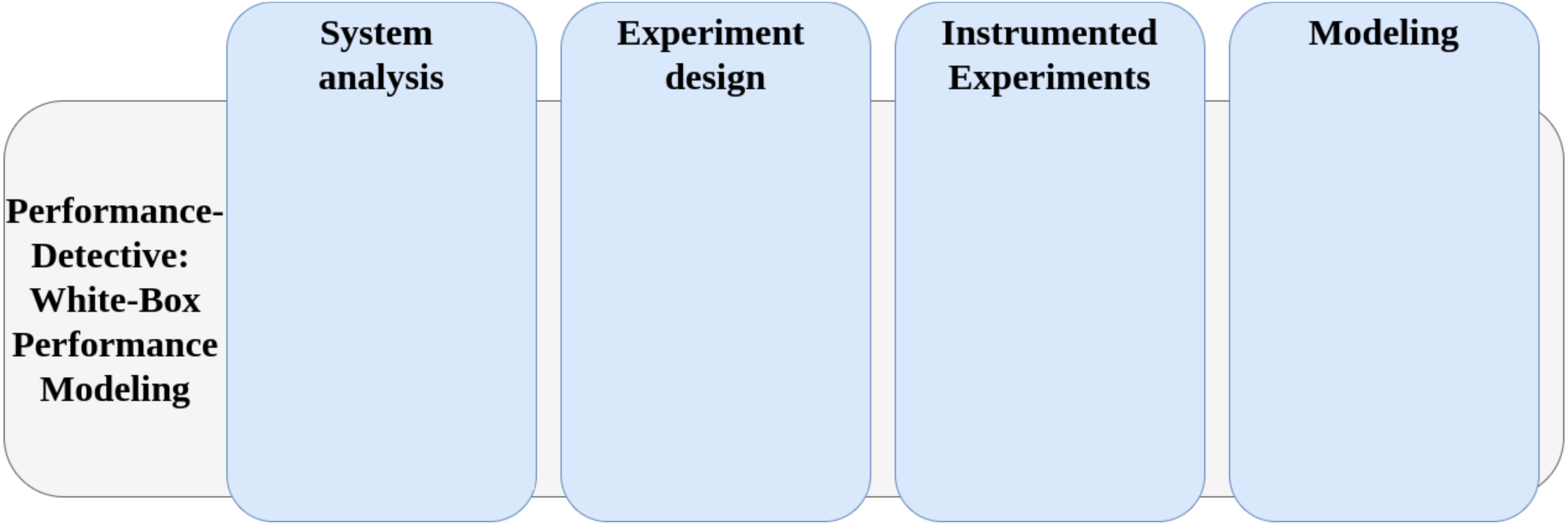
Current Modeling Workflows



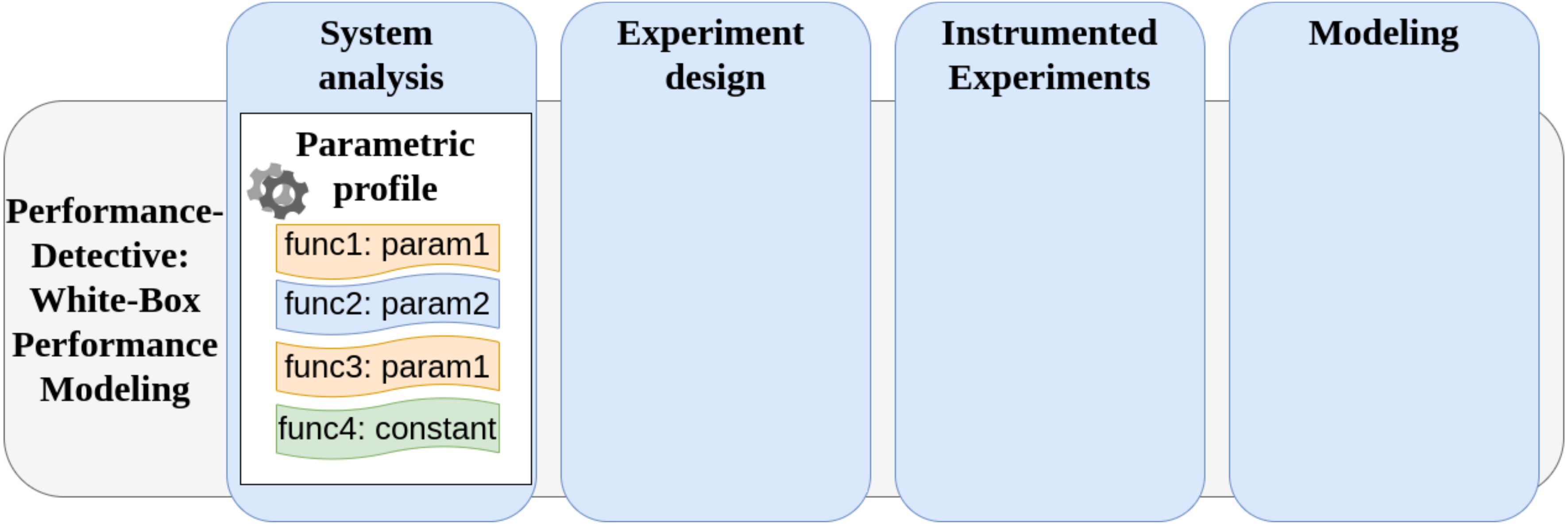
Current Modeling Workflows



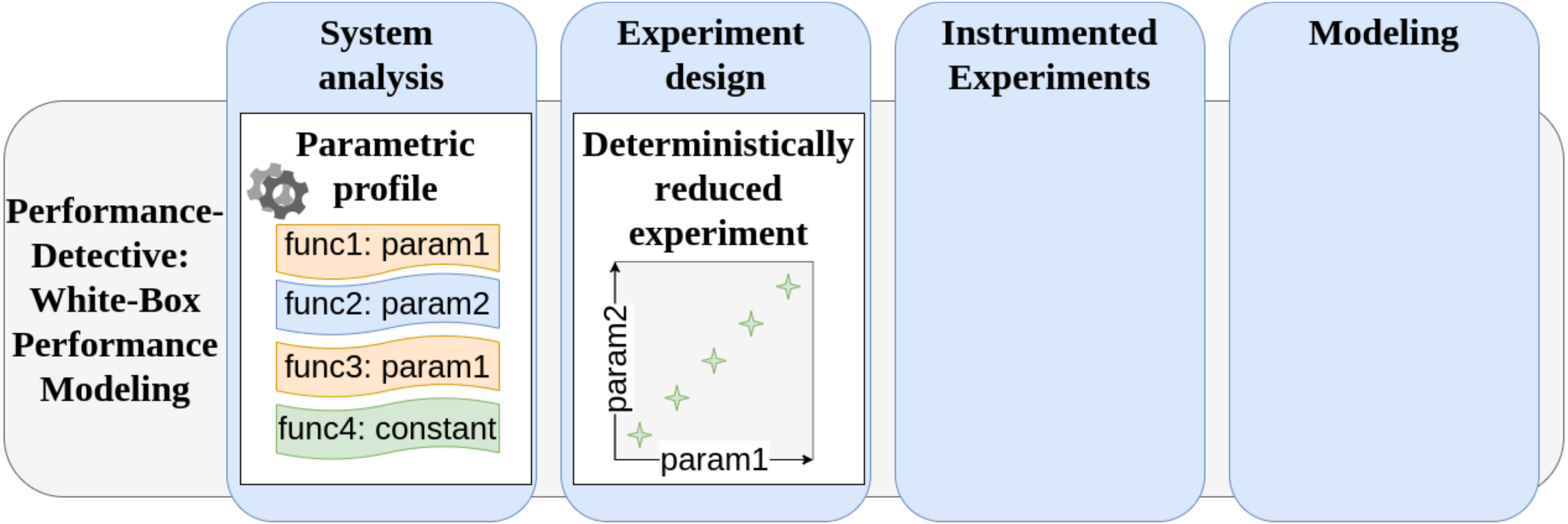
Performance-Detective Modeling Workflow



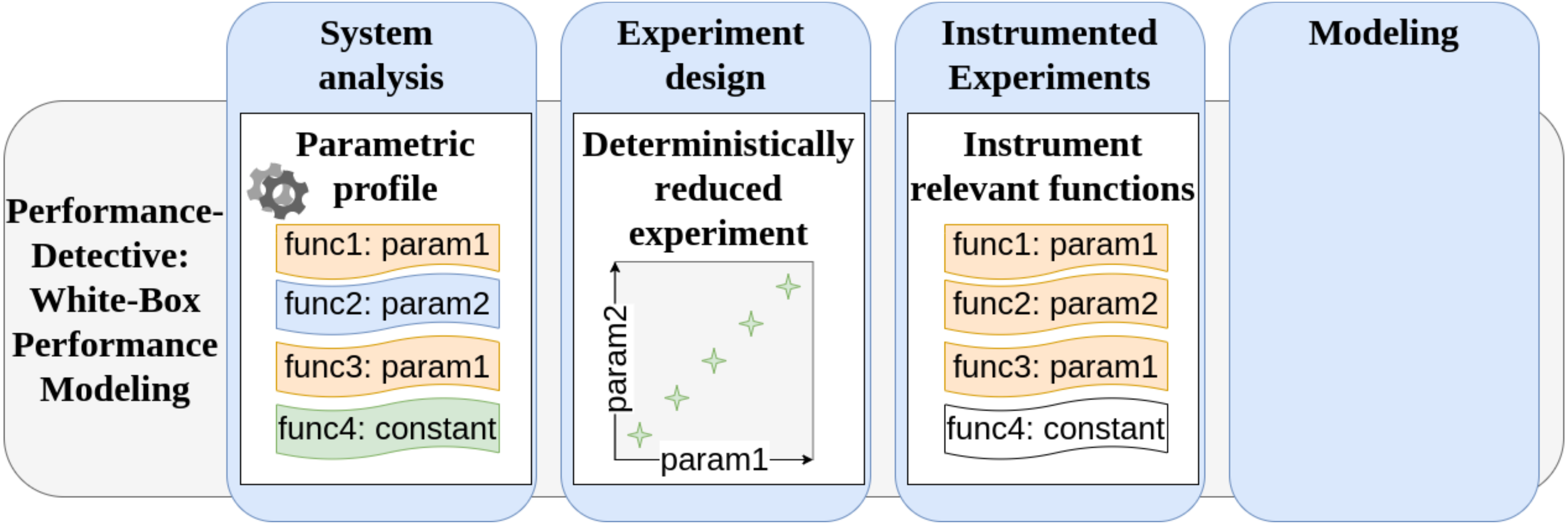
Performance-Detective Modeling Workflow



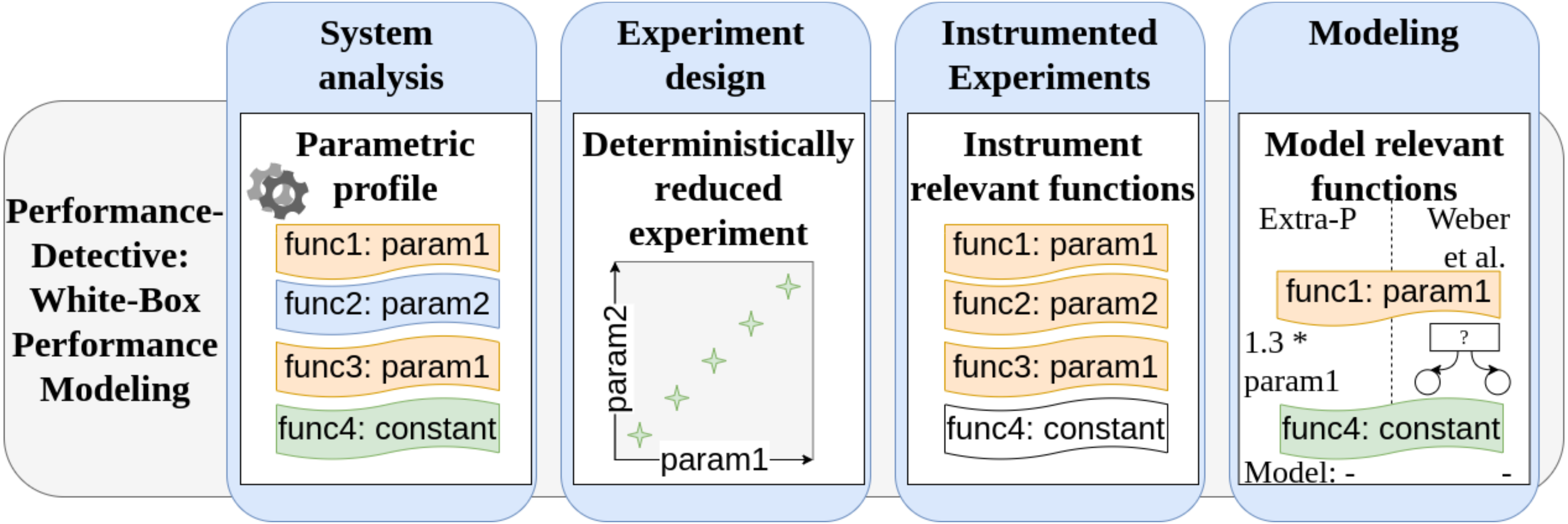
Performance-Detective Modeling Workflow



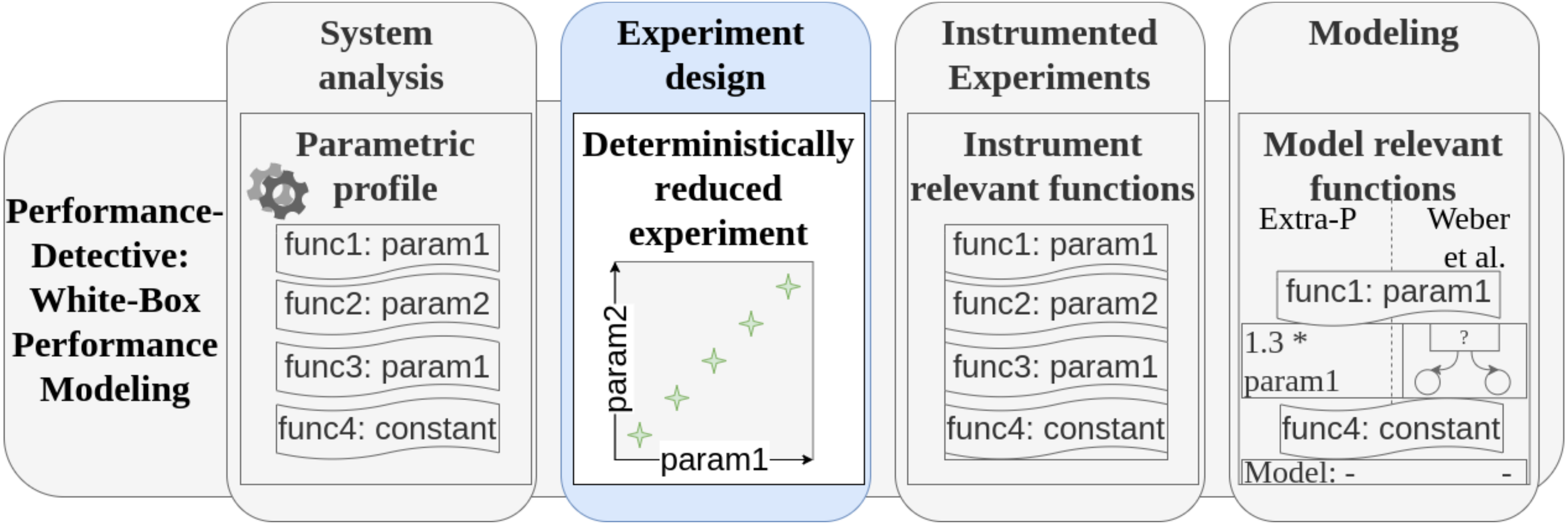
Performance-Detective Modeling Workflow



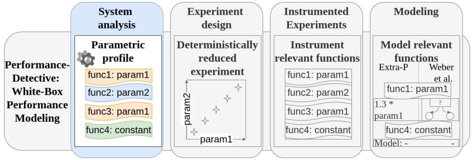
Performance-Detective Modeling Workflow



Performance-Detective Modeling Workflow



System analysis [Perf-Taint]



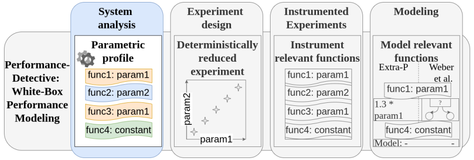
System: Parameters x1, x2, and iters

```
int main(int argc, char ** argv) {
  int x1 = atoi(argv[1]);
  int x2 = atoi(argv[2]);
  int iters = atoi(argv[3]);

  register_variables(x1, x2, iters)

  y = x1 * 5;
  z = x2 / 7;

  foo(y, z, iters);
}
```

y depends on x1

z depends on x2

System analysis [Perf-Taint]



Karlsruhe Institute of Technology

System: Parameters x1, x2, and iters

```
int main(int argc, char ** argv) {
  int x1 = atoi(argv[1]);
  int x2 = atoi(argv[2]);
  int iters = atoi(argv[3]);

  register_variables(x1, x2, iters)

  y = x1 * 5;
  z = x2 / 7;

  foo(y, z, iters);
}
```

y depends on x1

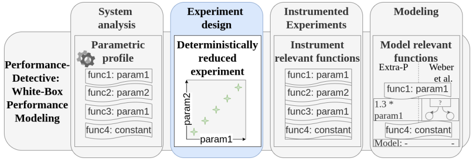
z depends on x2

```
void foo(int y, int z, int iters
  for (int i = 0; i < iters; i++
    for (int j = 0; j < y; j++) {
      //calculate something
    }
    bar(z);
  }
}

void bar(int z) {
  for (int i = 0; i < z; i++) {
    //calculate something
  }
  preCalculate(z)
}

int preCalculate(int z) {
  if (z < 10) return 0;
  return 1;
}
```

time(**foo**) = f(x1, iters)

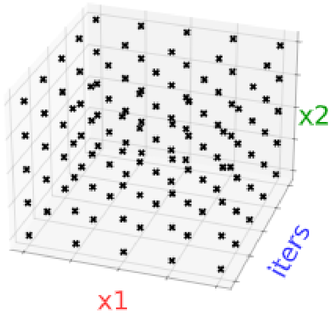
time(**bar**) = f(x2, iters)

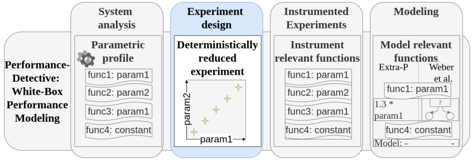
time(**preCalculate**) = constant



Experiment design

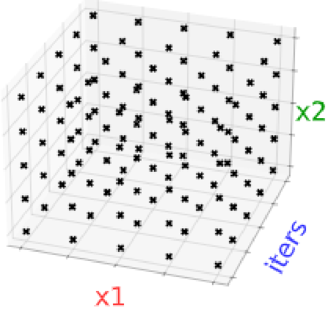
Full Experiment Design Space

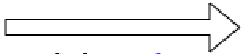




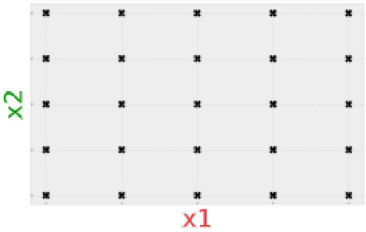
Experiment design

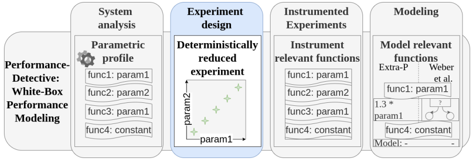
Full Experiment Design Space





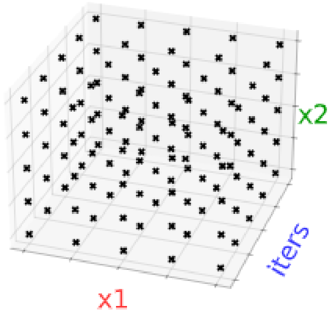
Insight 1: iters influences computations linearly



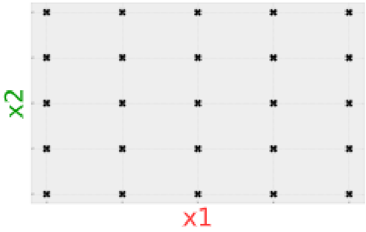


Experiment design

Full Experiment Design Space

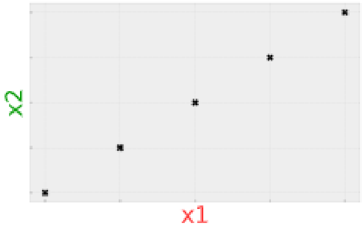


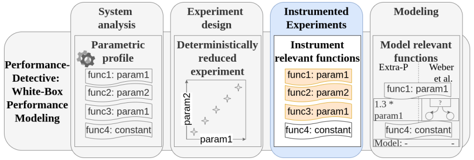
Insight 1: iters influences computations linearly



Insight 2: no function depending on **both** x1 and x2

Deduced Experiment Design





Profiling

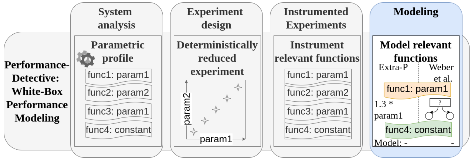
System: Parameters x1, x2, and iters

```
int main(int argc, char ** argv) {
  int x1 = atoi(argv[1]);
  int x2 = atoi(argv[2]);
  int iters = atoi(argv[3]);

  register_variables(x1, x2, iters)

  y = x1 * 5;
  z = x2 / 7;

  foo(y, z, iters);
}
```

```
void foo(int y, int z, int iters) {
  for (int i = 0; i < iters; i++) {
    for (int j = 0; j < y; j++) {
      //calculate something
    }
    bar(z);
  }
}

void bar(int z) {
  for (int i = 0; i < z; i++) {
    //calculate something
  }
  preCalculate(z)
}

int preCalculate(int z) {
  if (z < 10) return 0;
  return 1;
}
```

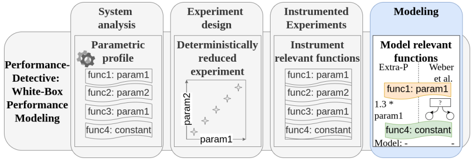
Skip repeated measurements

Modeling



- Performance-Detective is orthogonal to the instrumentation and learning methodology

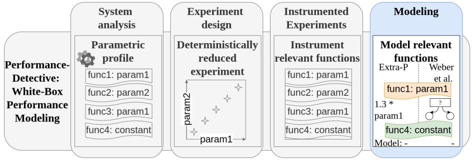
Approach	Sampling	Learning



Modeling

- Performance-Detective is orthogonal to the instrumentation and learning methodology

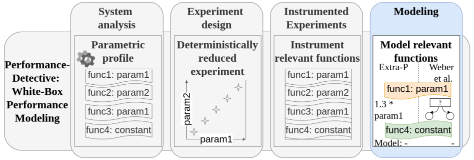
Approach	Sampling	Learning
Extra-P		
Performance-Influence Models		



Modeling

- Performance-Detective is orthogonal to the instrumentation and learning methodology

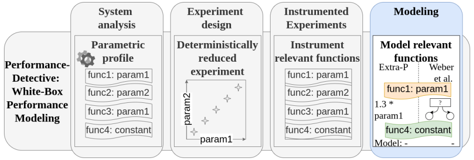
Approach	Sampling	Learning
Extra-P	Full-factorial	
	Sparse	
Performance-Influence Models		



Modeling

- Performance-Detective is orthogonal to the instrumentation and learning methodology

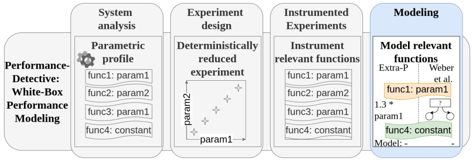
Approach	Sampling	Learning
Extra-P	Full-factorial	Regression to Performance-Model Normal Form
	Sparse	
Performance-Influence Models		



Modeling

- Performance-Detective is orthogonal to the instrumentation and learning methodology

Approach	Sampling	Learning
Extra-P	Full-factorial	Regression to Performance-Model Normal Form
	Sparse	
Performance-Influence Models	Plackett-Burman	



Modeling

- Performance-Detective is orthogonal to the instrumentation and learning methodology

Approach	Sampling	Learning
Extra-P	Full-factorial	Regression to Performance-Model Normal Form
	Sparse	
Performance-Influence Models	Plackett-Burman	Decision Trees

Case Studies

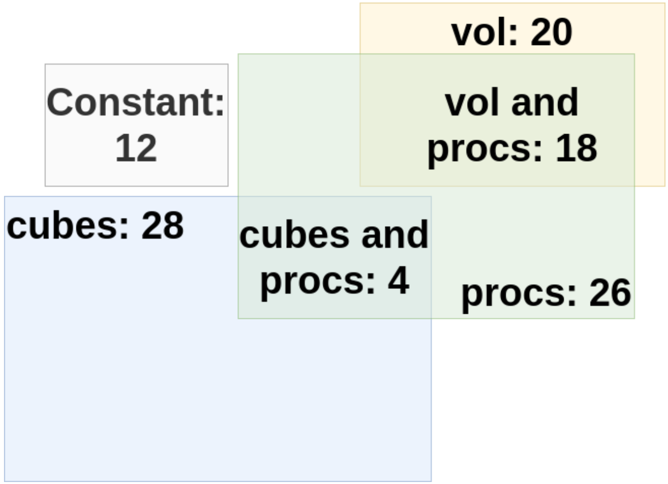
- Kripke: 3D Sn particle-transport application
- Pace3D: Multi-physics solver

Case Studies

- Kripke: 3D Sn particle-transport application
- **Pace3D: Multi-physics solver**

Pace3D: System analysis

- Pressure calculation with projected conjugate gradient method
- Parameters:
    - Number of processes – *procs*
    - Volume of material – *vol*
    - Spacing on the coarse grid – *cubes*
    - Iterations of the solver – *iters*



Pace3D: Experiment Design

Parameters:
vol - total volume
cubes - coarse grid size
procs - number of processes
iters - number of iterations

Full Experiment Design Space:
3125 experiment executions

- Full-factorial experiment design:
  - 5^4 different configurations
  - 5 repetitions per configuration

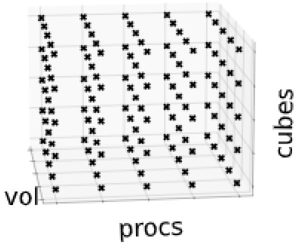
Pace3D: Experiment Design

Parameters:
vol - total volume
cubes - coarse grid size
procs - number of processes
iters - number of iterations

Full Experiment Design Space:
3125 experiment executions

Insight 1:
iters influences computations linearly

Full-factorial Experiment Design with 3 parameters:
125 experiment executions



Choose *iters* >= 5 instead of measurement repetitions

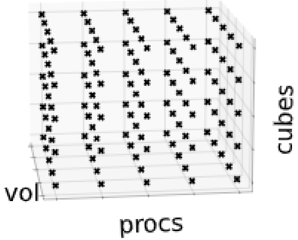
Pace3D: Experiment Design

Parameters:
vol - total volume
cubes - coarse grid size
procs - number of processes
iters - number of iterations

Full Experiment Design Space:
3125 experiment executions

Insight 1:
iters influences computations linearly

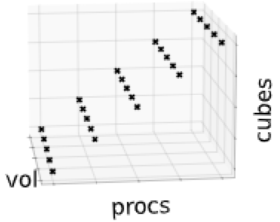
Full-factorial Experiment Design with 3 parameters:
125 experiment executions



Choose *iters* >= 5 instead of measurement repetitions

Insight 2:
vol and *cubes* do not interact

Deterministically reduced Experiment Design:
25 experiment executions



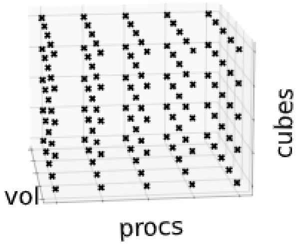
Pace3D: Experiment Design

Parameters:
vol - total volume
cubes - coarse grid size
procs - number of processes
iters - number of iterations

Full Experiment Design Space:
3125 experiment executions

Insight 1:
iters influences computations linearly

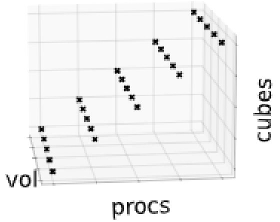
Full-factorial Experiment Design with 3 parameters:
125 experiment executions



Choose *iters* >= 5
instead of measurement repetitions

Insight 2:
vol and *cubes* do not interact

Deterministically reduced Experiment Design:
25 experiment executions



25 instead of 3125 experiment executions – 4% of samples needed.

RQ1: Modeling using a single repetition of experiments

Modeling approach	#repetitions	Mean error Pace3D		Mean error Kripke	
		interpolate	extrapolate	interpolate	extrapolate
Extra-P	5	8.3%	9.3%		

RQ1: Modeling using a single repetition of experiments

Modeling approach	#repetitions	Mean error Pace3D		Mean error Kripke	
		interpolate	extrapolate	interpolate	extrapolate
Extra-P	5	8.3%	9.3%		
Extra-P	**1**	**8.1%**	**8.7%**		

RQ1: Modeling using a single repetition of experiments

Modeling approach	#repetitions	Mean error Pace3D		Mean error Kripke	
		interpolate	extrapolate	interpolate	extrapolate
Extra-P	5	8.3%	9.3%	4.6%	18.3%
Extra-P	**1**	**8.1%**	**8.7%**		

RQ1: Modeling using a single repetition of experiments

Modeling approach	#repetitions	Mean error Pace3D		Mean error Kripke	
		interpolate	extrapolate	interpolate	extrapolate
Extra-P	5	8.3%	9.3%	4.6%	18.3%
Extra-P	**1**	**8.1%**	**8.7%**	**4.3%**	**15.2%**

RQ1: Modeling using a single repetition of experiments

Modeling approach	#repetitions	Mean error Pace3D		Mean error Kripke	
		interpolate	extrapolate	interpolate	extrapolate
Extra-P	5	8.3%	9.3%	4.6%	18.3%
Extra-P	**1**	**8.1%**	**8.7%**	**4.3%**	**15.2%**
PIM	5	21.6%	60.2%		
PIM	**1**				

RQ1: Modeling using a single repetition of experiments

Modeling approach	#repetitions	Mean error Pace3D		Mean error Kripke	
		interpolate	extrapolate	interpolate	extrapolate
Extra-P	5	8.3%	9.3%	4.6%	18.3%
Extra-P	**1**	**8.1%**	**8.7%**	**4.3%**	**15.2%**
PIM	5	21.6%	60.2%		
PIM	**1**	**22.6%**	**62.8%**		

RQ1: Modeling using a single repetition of experiments

Karlsruhe Institute of Technology

Modeling approach	#repetitions	Mean error Pace3D		Mean error Kripke	
		interpolate	extrapolate	interpolate	extrapolate
Extra-P	5	8.3%	9.3%	4.6%	18.3%
Extra-P	**1**	**8.1%**	**8.7%**	**4.3%**	**15.2%**
PIM	5	21.6%	60.2%	21.8%	25.7%
PIM	**1**	**22.6%**	**62.8%**		

RQ1: Modeling using a single repetition of experiments

Modeling approach	#repetitions	Mean error Pace3D		Mean error Kripke	
		interpolate	extrapolate	interpolate	extrapolate
Extra-P	5	8.3%	9.3%	4.6%	18.3%
Extra-P	**1**	**8.1%**	**8.7%**	**4.3%**	**15.2%**
PIM	5	21.6%	60.2%	21.8%	25.7%
PIM	**1**	**22.6%**	**62.8%**	**22.4%**	**25.3%**

RQ2: Varying independent parameters simultaneously – Modeling with Extra-P

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓			
Full-factorial	✓			
Full-factorial	-			

RQ2: Varying independent parameters simultaneously – Modeling with Extra-P

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**		
Full-factorial	✓	367.6		
Full-factorial	-	367.6		

RQ2: Varying independent parameters simultaneously – Modeling with Extra-P

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**	**8.1%**	**8.7%**
Full-factorial	✓	367.6		
Full-factorial	-	367.6		

RQ2: Varying independent parameters simultaneously – Modeling with Extra-P

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**	**8.1%**	**8.7%**
Full-factorial	✓	367.6	9.5%	10.7%
Full-factorial	-	367.6		

RQ2: Varying independent parameters simultaneously – Modeling with Extra-P

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**	**8.1%**	**8.7%**
Full-factorial	✓	367.6	9.5%	10.7%
Full-factorial	-	367.6	6.2%	6.3%

RQ2: Varying independent parameters simultaneously – Modeling with Extra-P

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓			
Sparse	✓			
Sparse	-			

RQ2: Varying independent parameters simultaneously – Modeling with Extra-P

Karlsruhe Institute of Technology

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**		
Sparse	✓	5.5		
Sparse	-	5.5		

RQ2: Varying independent parameters simultaneously – Modeling with Extra-P

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**	**8.1%**	**8.7%**
Sparse	✓	5.5		
Sparse	-	5.5		

RQ2: Varying independent parameters simultaneously – Modeling with Extra-P

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**	**8.1%**	**8.7%**
Sparse	✓	5.5	8.9%	17.7%
Sparse	-	5.5		

RQ2: Varying independent parameters simultaneously – Modeling with Extra-P

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**	**8.1%**	**8.7%**
Sparse	✓	5.5	8.9%	17.7%
Sparse	-	5.5	15.0%	31.8%

RQ2: Varying independent parameters simultaneously – Modeling with Decision Trees

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓			
Plackett-Burman	✓			
Plackett-Burman	-			

RQ2: Varying independent parameters simultaneously – Modeling with Decision Trees

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**		
Plackett-Burman	✓	164.4		
Plackett-Burman	-	164.4		

RQ2: Varying independent parameters simultaneously – Modeling with Decision Trees

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**	**22.6%**	**47.7%**
Plackett-Burman	✓	164.4		
Plackett-Burman	-	164.4		

RQ2: Varying independent parameters simultaneously – Modeling with Decision Trees

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**	**22.6%**	**47.7%**
Plackett-Burman	✓	164.4	20.1%	45.4%
Plackett-Burman	-	164.4		

RQ2: Varying independent parameters simultaneously – Modeling with Decision Trees

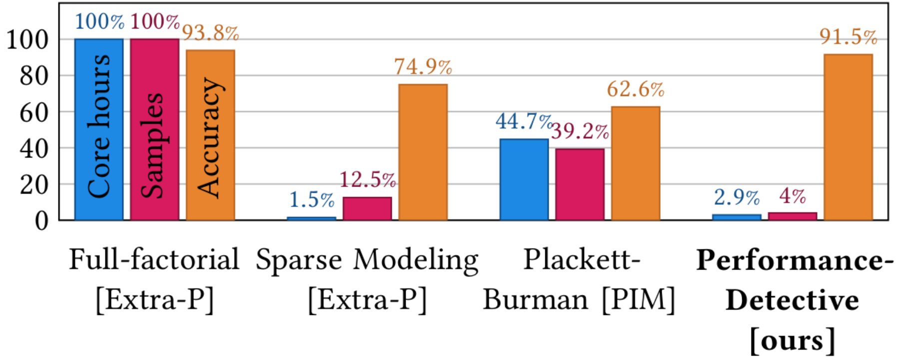
Karlsruhe Institute of Technology

Experiment Design	Perf-Taint	Cost in core hours	Error	
			interpolated	extrapolated
Performance-Detective	✓	**10.9**	**22.6%**	**47.7%**
Plackett-Burman	✓	164.4	20.1%	45.4%
Plackett-Burman	-	164.4	27.0%	44.2%

Conclusion

- **Problem**: Performance modeling workflows use heuristic sampling
- **Contribution**: Deduce minimal necessary experiment design
- Evaluation on two applications shows maintained model accuracy while significantly reducing needed core hours

Results for Pace3D



Results for Kripke