Determining Levels of Detail for Simulators of Parallel and Distributed Computing Systems via Automated Calibration

Jesse McDonald[1] Yick-Ching (Jeff) Wong[1] Kshitij Mehta[2] Frédéric Suter[2] Rafael Ferreira da Silva[2] Loïc Pottier[3] Ewa Deelman[4] Henri Casanova[1]

[1]University of Hawai'i at Mānoa, U.S.A.

[2]Oak Ridge National Laboratory, U.S.A.

[3]Lawrence Livermore National Laboratory, U.S.A.

[4]University of Southern California, U.S.A.

PDC Simulation

- Simulation is used extensively in Parallel and Distributed Computing (PDC) research
  - Repeatable, fully observable
  - Can experiment with arbitrary platform/application configurations
  - Often less time-, labor-, and cost-intensive than real-world experiments
- Simulation models abstract away real-world behavior
- Requires accurate simulator

Sources of Simulator inaccuracy

Main sources of Simulator Error

1. Incorrect parameter values
    - Simulators are built to simulate anything
    - Are you simulating what you intend?
2. Insufficient Simulation models (Sophistication)
    - How complex do simulation models need to be?
    - Is the model being used even capable of reaching the desired level of detail?

Challenge: If the simulator is not sufficiently accurate, why?

- Maybe its parameters are bad? (Insufficient Calibration)
- Maybe its models are bad? (Insufficient Sophistication)

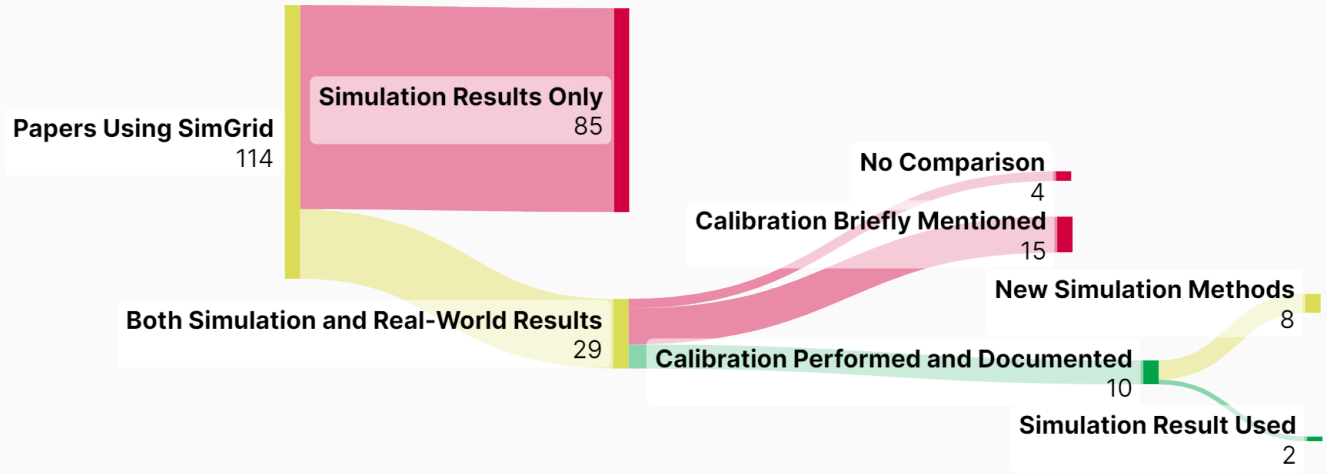
Simulation Calibration

Solution to bad Parameters: Simulation Calibration

- Pick some metric you care about
- Pick some ground-truth data to compare to
- Apply metric to simulation runs and Ground-truth data to quantify error (loss function)
- Pick parameters that minimize that error

State of Simulation Calibration

- Sounds simple: What are other researchers doing?
  114 papers published in 2017-2022 that include results obtained with the SimGrid simulation framework



Finding: Only two papers that do not focus on simulation as the research result perform and document a sound calibration

- Those two describe a tedious manual process

Bottom line: Calibration should be automated

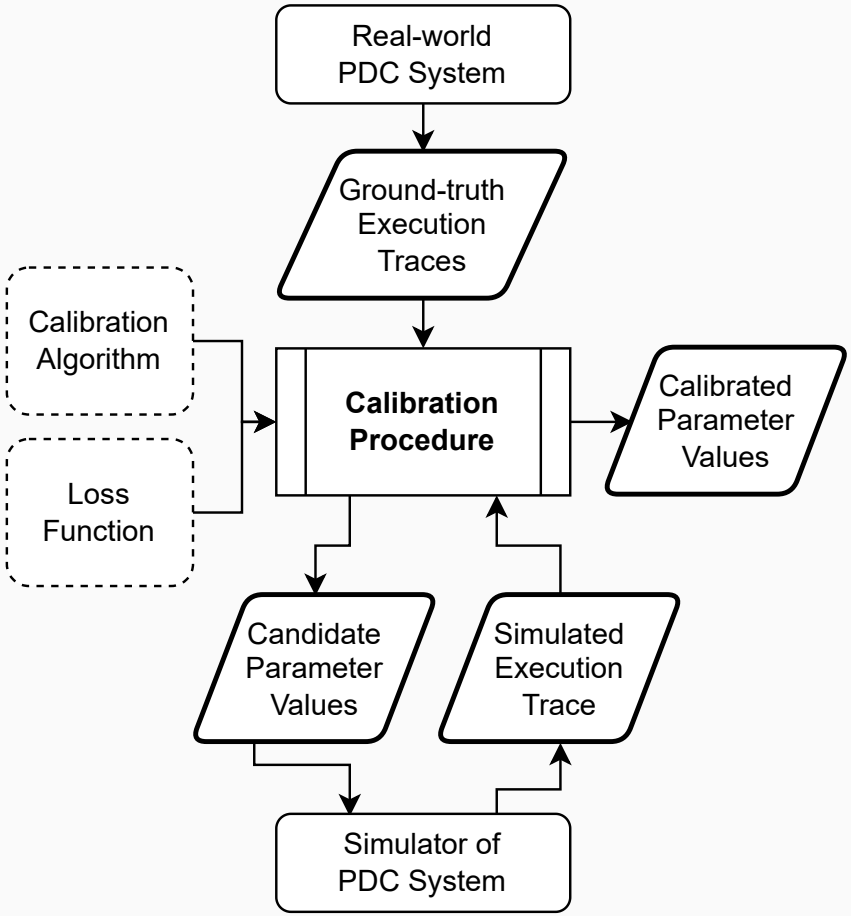
Step 1: Automate Calibration

Main Problems:

- What loss function to use?
- How to solve the optimization problem?
- What ground truth to use?

Other Complications:

- Simulators are non-differentiable
- Simulators are non-continuous
- Simulators are slow to sample



Loss Function

What can we use?

- Makespan?
- Job execution times?
- Platform utilization?
- Something else?

What should we use?

- Do simple loss functions allow too many Degrees of Freedom?
- Does unimportant data in complex loss functions overpower important features?
- How do we even compare two loss functions?

Synthetic Calibration

- Pick some set of loss functions
- Pick an arbitrary "Calibration"
- Generate fake ground-truth data using that calibration
- Run automated simulation calibration with each loss function
- Use "relative L_1" error to measure parameter error

$$D(C, T) = \sum_{i=1}^{\dim(\mathbf{T})} \left| \frac{C_i - T_i}{T_i} \right|$$

- Pick loss function with lowest parameter error

Optimization Algorithm

- Many options
- We considered:
  - Random Search
  - Grid Search – Eliminated in preliminary experiments
  - Gradient Descent with Backtracking Line Search – Eliminated in preliminary experiments
  - Baysian Optimization – 4 Implementations in skopt
    - All 4 implementations performed identically in preliminary
    - Proceed with Gaussian Processes Only
- How well each works depends on the loss function used
  - Select optimization algorithm at same time as loss function
- Each can take a different amount of time/simulations

Ground-truth Data

- Expensive to collect
- Too little is bad
- How much is needed?
- What type of data?

Case Studies

Testing our Methodology

1. Simulator of Workflow executions on Chameleon Cloud
2. Simulator of MPI Application Executions on Summit

Using our python framework SimCal

- Supports Arbitrary Simulators and Loss Functions
- Supports Arbitrary Optimization Algorithms
- Implements multi-core parallelism

Equal Time Calibration

- Different simulators take different times
- Different Loss functions take different time
- We allocate the same amount of time to each calibration
- Allows fair comparison of methods

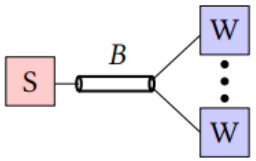
Chameleon Cloud Workflow Simulator

- Written in C++ using WRENCH (which uses SimGrid)
- Input:
  - Workflow
  - Platform
- Outputs:
  - Task execution times/locations
  - Overall Makespan
- Unknown Platform Specifics
- Sophistication Options
  - Network Configurations
  - Storage Configurations
  - Workflow Management System

Possible Configurations

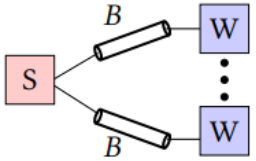
Network topology simulation options

One-link topology



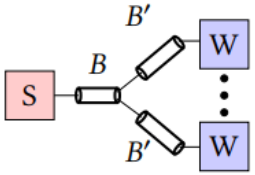
B: bandwidth

Star topology



B: bandwidth

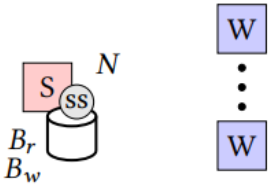
One-link-plus-star topology



B: bandwidth
B': bandwidth

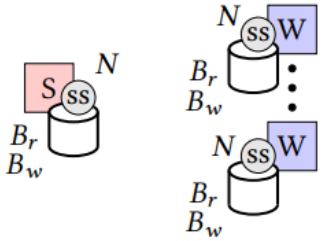
Storage system simulation options

Storage service (SS) on submit node only



B_r: read bandwidth
B_w: write bandwidth
N: max # concurrent disk reads/writes

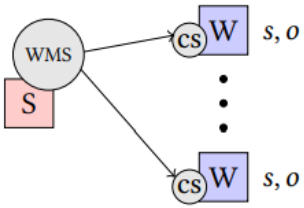
Storage service (SS) on submit and worker nodes



B_r: read bandwidth
B_w: write bandwidth
N: max # concurrent disk reads/writes

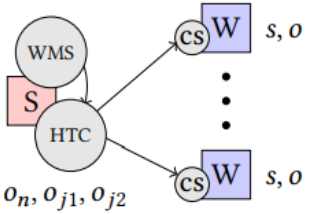
Compute system simulation options

WMS accesses compute services (CS) directly



s: core speed
o: task startup overhead

WMS accesses compute services (CS) via HTCondor (HTC)



s: core speed
o: task startup overhead
o_n: HTCondor negotiator overhead
o_{j1}: HTCondor job pre-overhead
o_{j2}: HTCondor job post-overhead

Parameters

- We select which parameters to calibrate based on platform knowledge
  - Thousands of small control message parameters
  - Platform is fast, low latency
  - Assume control messages negligible
  - Assume platform is homogeneous
- We pick wide ranges
  - Exponential $2^{20} - 2^{40}$ for speeds/bandwidths
  - Linear 0-10ms for latencies
- Any correct calibration SHOULD be in this range

Groundtruth

Workflow Application	Workflow Size (#tasks)[1]	Work / Task (sec.)[2]	Data Footprint (MB)[3]
Epigenomics (bioinformatics)	43, 64, 86, 129, 215	0.6, 1.15, 1.73, 7.22, 73.25	0, 150, 1500, 15000
1000Genome (bioinformatics)	54, 81, 108, 162, 270	0.9, 1.47, 2.11, 8.02, 80.94	0, 150, 1500, 15000
SoyKB (bioinformatics)	98, 147, 196, 294, 490	0.53, 1.06, 1.6, 6.55, 74.21	0, 150, 1500, 15000
Montage (astronomy)	60, 90, 120, 180, 300	0.59, 1.12, 1.75, 7.07, 73.13	0, 150, 1500, 15000
Seismology (seismology)	103, 154, 206, 309, 515	0.74, 1.28, 1.91, 8.34, 86.25	0, 150, 1500, 15000
Chain (synthetic)	10, 25, 50	0.83, 1.36, 1.85, 5.74, 48.94	0, 150, 1500
Forkjoin (synthetic)	10, 25, 50	0.84, 1.39, 2.05, 7.61, 70.76	0, 150, 1500

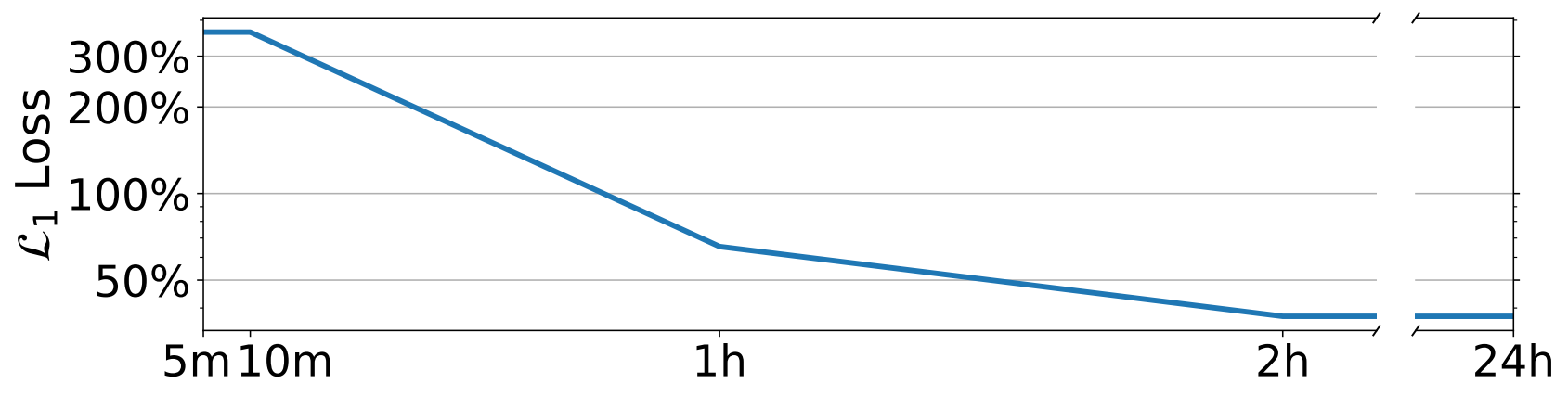
- 5 repeats each
- 9200 workflow executions
- Chameleon Cloud 48 core intel icelake processors with Pegasus and HTCondor

Loss Functions

- $\mathcal{L}_1$: $\text{avg}_i(e_i)$
- $\mathcal{L}_2$: $\max_i(e_i)$
- $\mathcal{L}_3$: $\text{avg}_i(e_i + \text{avg}_j(e_{i,j}))$
- $\mathcal{L}_4$: $\max_i(e_i + \text{avg}_j(e_{i,j}))$
- $\mathcal{L}_5$: $\text{avg}_i(e_i + \max_j(e_{i,j}))$
- $\mathcal{L}_6$: $\max_i(e_i + \max_j(e_{i,j}))$
- workflow i
- m_i ground-truth, $\hat{m}_i$ simulated makespan
- $t_{i,j}$ ground-truth, $\hat{t}_{i,j}$ simulated task execution time (task j)
- $e_i = |(m_i - \hat{m}_i)/m_i|$
- $e_{i,j} = |(t_{i,j} - \hat{t}_{i,j})/t_{i,j}|$
- More possible

Other Details

- 24 Hour Time Budget



- Calibrated on:
  - University of Hawai‘i at Mānoa’s Koa cluster
  - Dedicated Intel Xeon 48 core CPU per calibration

Synthetic Benchmarking

- $\mathcal{L}_1$: $\text{avg}_i(e_i)$
- $\mathcal{L}_2$: $\max_i(e_i)$
- $\mathcal{L}_3$: $\text{avg}_i(e_i + \text{avg}_j(e_{i,j}))$
- $\mathcal{L}_4$: $\max_i(e_i + \text{avg}_j(e_{i,j}))$
- $\mathcal{L}_5$: $\text{avg}_i(e_i + \max_j(e_{i,j}))$
- $\mathcal{L}_6$: $\max_i(e_i + \max_j(e_{i,j}))$

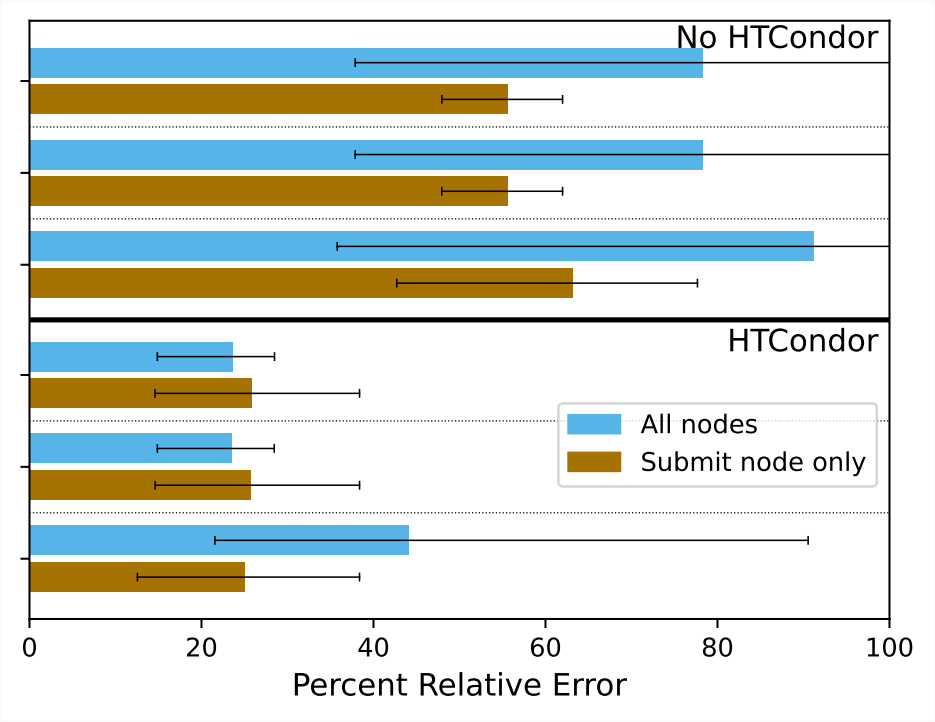
Alg. \ Loss	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_4$	$\mathcal{L}_5$	$\mathcal{L}_6$
RAND	541.24	111.43	610.82	883.40	130.55	883.40
BO-GP	**30.96**	935.10	935.10	89.76	89.76	89.76

- BO-GP with $\mathcal{L}_1$ best performing

Sophistication Setup

- Ground-Truth Data: 2nd most workers/tasks only
- Theoretically good data with test data beyond the data
- Calibrated 12 versions
- For 24 hours each
- With $\mathcal{L}_1$
- With BO-GP
- on Koa

Sophistication Result



- Simulate HTCondor
- Simulate Simple Network
  - Complex network theoretically can mimic either
  - In reality, just more parameters
- Storage location mostly irrelevant
- 20% error

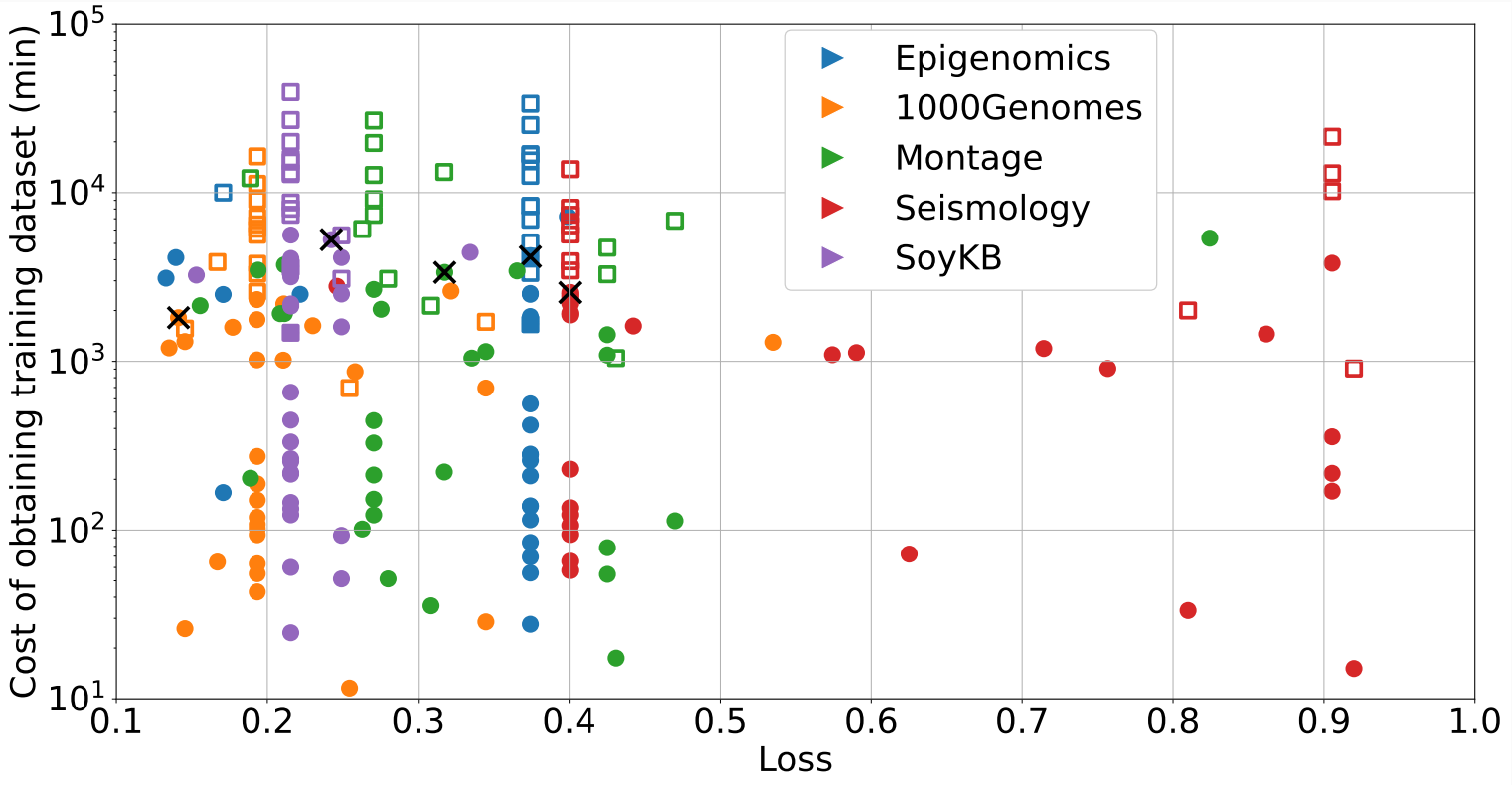
State-of-the-Art Methodology Comparison

- Attempted Manual Calibration
- Parameters Based on discovery
- Simple Simulator error: 110% - 1400% depending on workflow
- Complex simulator: 60% - 110% depending on workflow
- Conclusion: Our method is a significant improvement

Impact of Ground-Truth Data

- But did we pick good ground-truth data?
- Using best simulator, BO-GP, 24h, $\mathcal{L}_1$
- Testing on All Extreme Data
- Trained with various data subsets
- Quantified Compute Resources to acquire each data subset

Ground-Truth Result



- More data not always better
- Diversity needed

Benchmark Only

- Calibrated using Chain and Forkjoin
- Tested on other workflows
- Didnt really work
- Forkjoin only did best, barely better than human
- Forkjoin+Chain slightly worse
- Chain only did very bad, worse than human

Case Study 2: SMPI

- Simulated MPI applications on Summit
- Intel MPI Benchmark Suite
- All details in the paper

Result: Calibration methodology works: Identified inherent flaws in the ground-truth data

Conclusion

- Automated Calibration is a good way to reduce error due to parameter values
    - Which then allows comparing simulator versions implemented at different levels of detail
- Ground-truth data should be diverse
    - But more is not necessarily better
- Synthetic Calibration allows sound comparison of different loss functions

Questions?

Simgrid

simgrid.org/

WRENCH

wrench-project.org/

SimCal

github.com/
wrench-project/simcal