How Predictable are HPC Applications' I/O (and why should we care)

A Case for Smarter I/O and Storage Systems

Matthieu Dorier
Argonne National Laboratory

Argonne NATIONAL LABORATORY

Let's take the machine's point of view, shall we?

What can the machine know about the applications?

```
void output(const char* filename,…)
{
    H5Fopen(filename,…);
    …
    H5Dwrite(…);
    H5Dwrite(…);
    …
    H5Fclose(…);
}

int main(int argc, char** argv) {
    …
    while(not done) {
        solve(…);
        output(“checkpoint.h5”,…);
        output(“analysis.h5”,…);
    }
}
```

```
fopen “checkpoint.h5” -> fd=42
fwrite fd=42 size=1024 count=8
fwrite fd=42 size=2048 count=8
fwrite fd=42 size=1024 count=8
fwrite fd=42 size=512 count=4
fwrite fd=42 size=512 count=4
fwrite fd=42 size=512 count=4
fclose fd=42
fopen “analysis.h5” -> fd=43
fwrite fd=43 size=1024 count=8
fwrite fd=43 size=128 count=4
fwrite fd=43 size=1024 count=8
fwrite fd=43 size=20148 count=8
fclose fd=43
fopen “checkpoint.h5” -> fd=44
fwrite fd=44 size=1024 count=8
fwrite fd=44 size=2048 count=8
fwrite fd=44 size=1024 count=8
fwrite fd=44 size=512 count=4
fwrite fd=44 size=512 count=4
fwrite fd=44 size=512 count=4
fclose fd=44
```

Modeling a Sequence of Events

Modeling Randomness

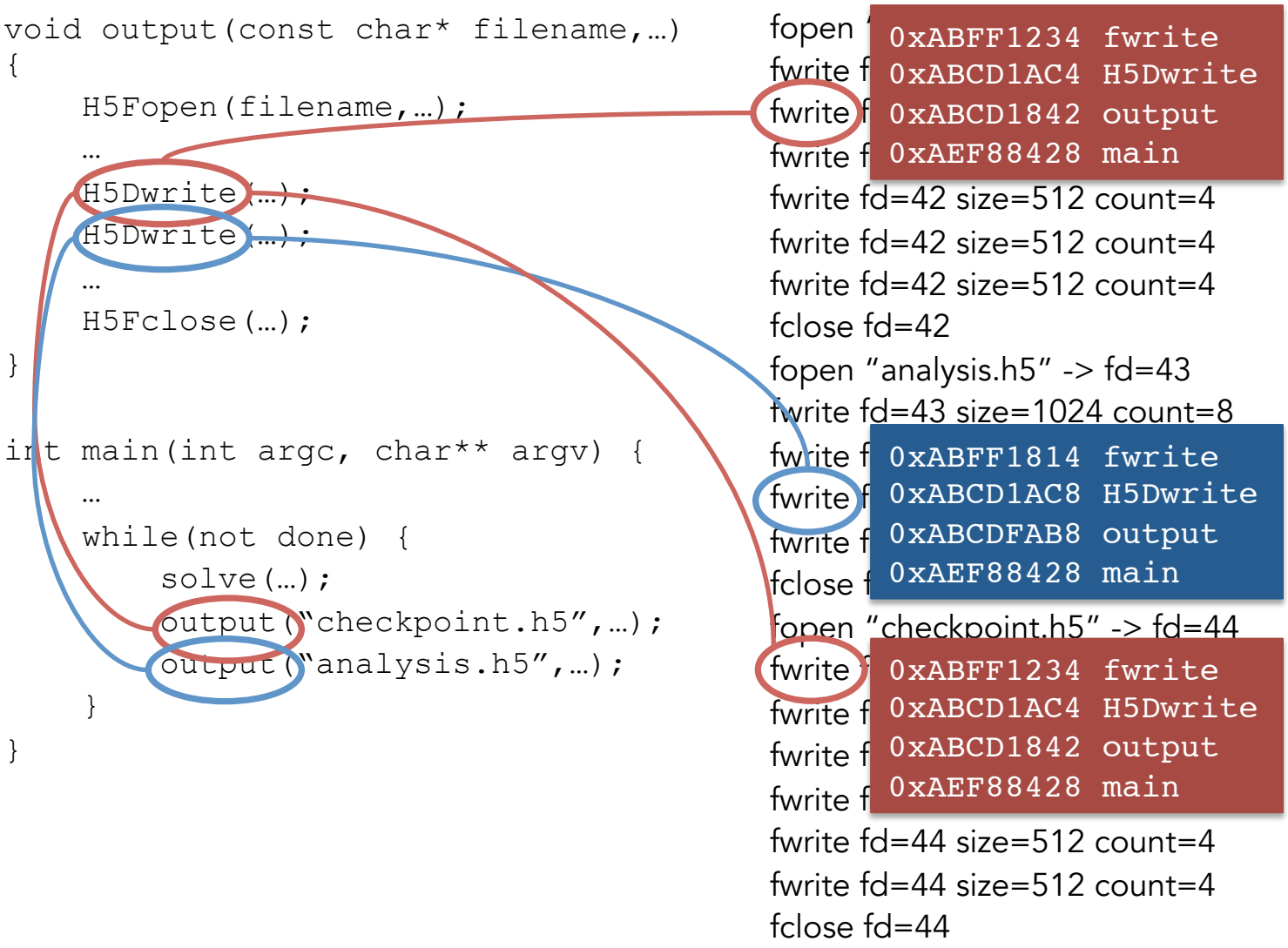
"what kind of distribution does it follow?"
"what kind of stochastic process drives it?"

Modeling Determinism

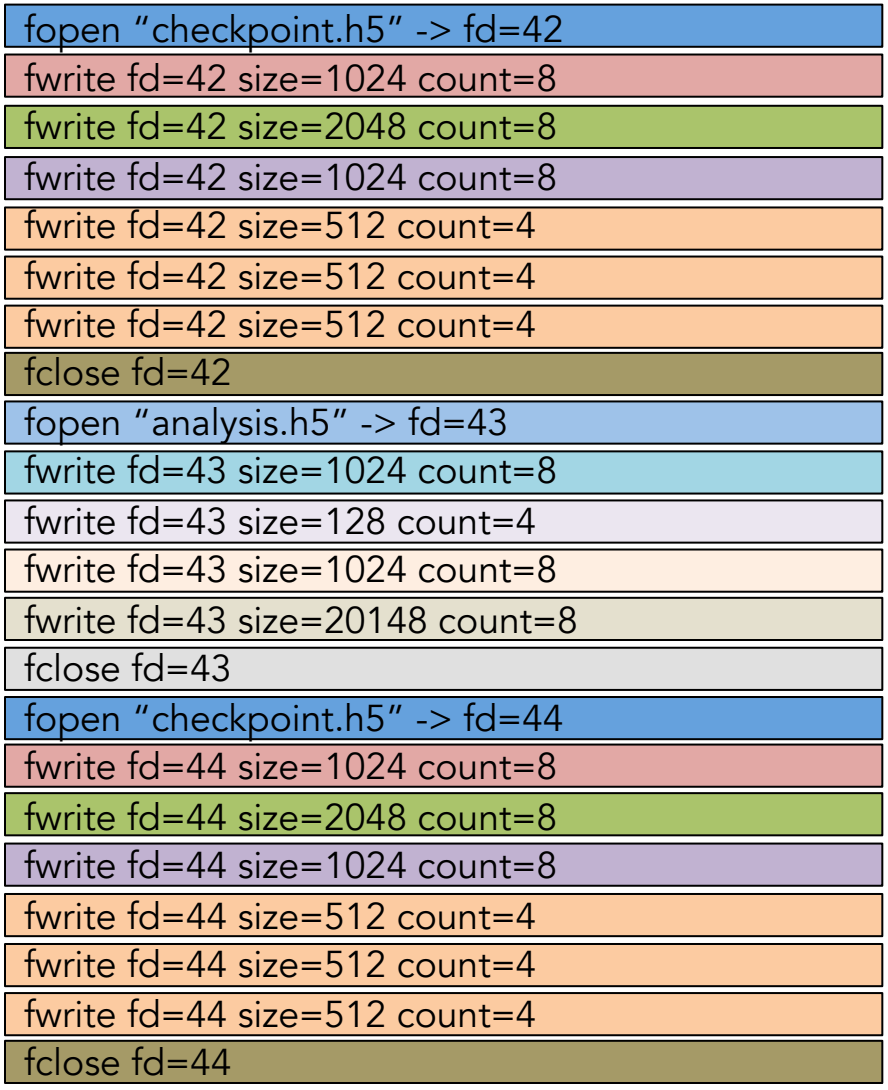
"what rules does it follow?"
"what is the logic behind it?"

ARIMA models
Markov models
Statistics

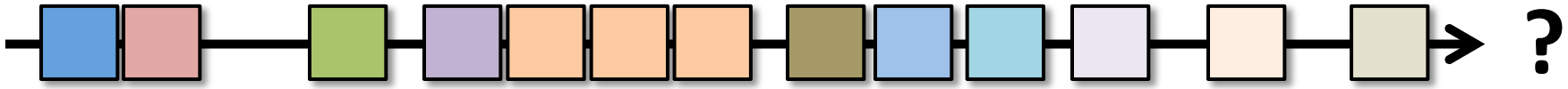
Compression
Formal Languages

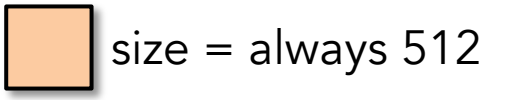

void output(const char* filename,…)
{
H5Fopen(filename,…);
…
H5Dwrite(…);
H5Dwrite(…);
…
H5Fclose(…);
}
int main(int argc, char** argv) {
…
while(not done) {
solve(…);
output("checkpoint.h5",…);
output("analysis.h5",…);
}
}
0xABFF1234 fwrite
0xABCD1AC4 H5Dwrite
0xABCD1842 output
0xAEF88428 main
0xABFF1814 fwrite
0xABCD1AC8 H5Dwrite
0xABCDFAB8 output
0xAEF88428 main
0xABFF1234 fwrite
0xABCD1AC4 H5Dwrite
0xABCD1842 output
0xAEF88428 main
fwrite fd=42 size=512 count=4
fwrite fd=42 size=512 count=4
fwrite fd=42 size=512 count=4
fclose fd=42
fopen "analysis.h5" -> fd=43
fwrite fd=43 size=1024 count=8
fwrite fd=44 size=512 count=4
fwrite fd=44 size=512 count=4
fclose fd=44

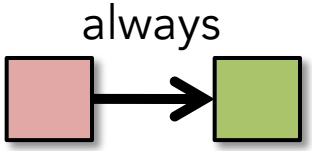
```
void output(const char* filename,…)
{
    H5Fopen(filename,…);
    …
    H5Dwrite(…);
    H5Dwrite(…);
    …
    H5Fclose(…);
}

int main(int argc, char** argv) {
    …
    while(not done) {
        solve(…);
        output(“checkpoint.h5”,…);
        output(“analysis.h5”,…);
    }
}
```

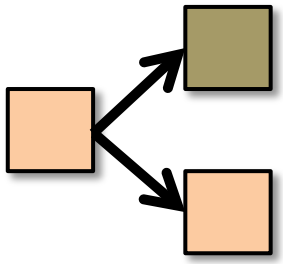


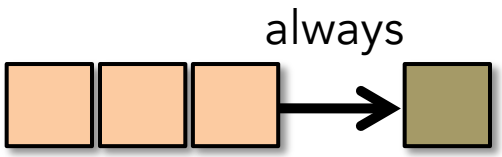
Build Rules, Build Models



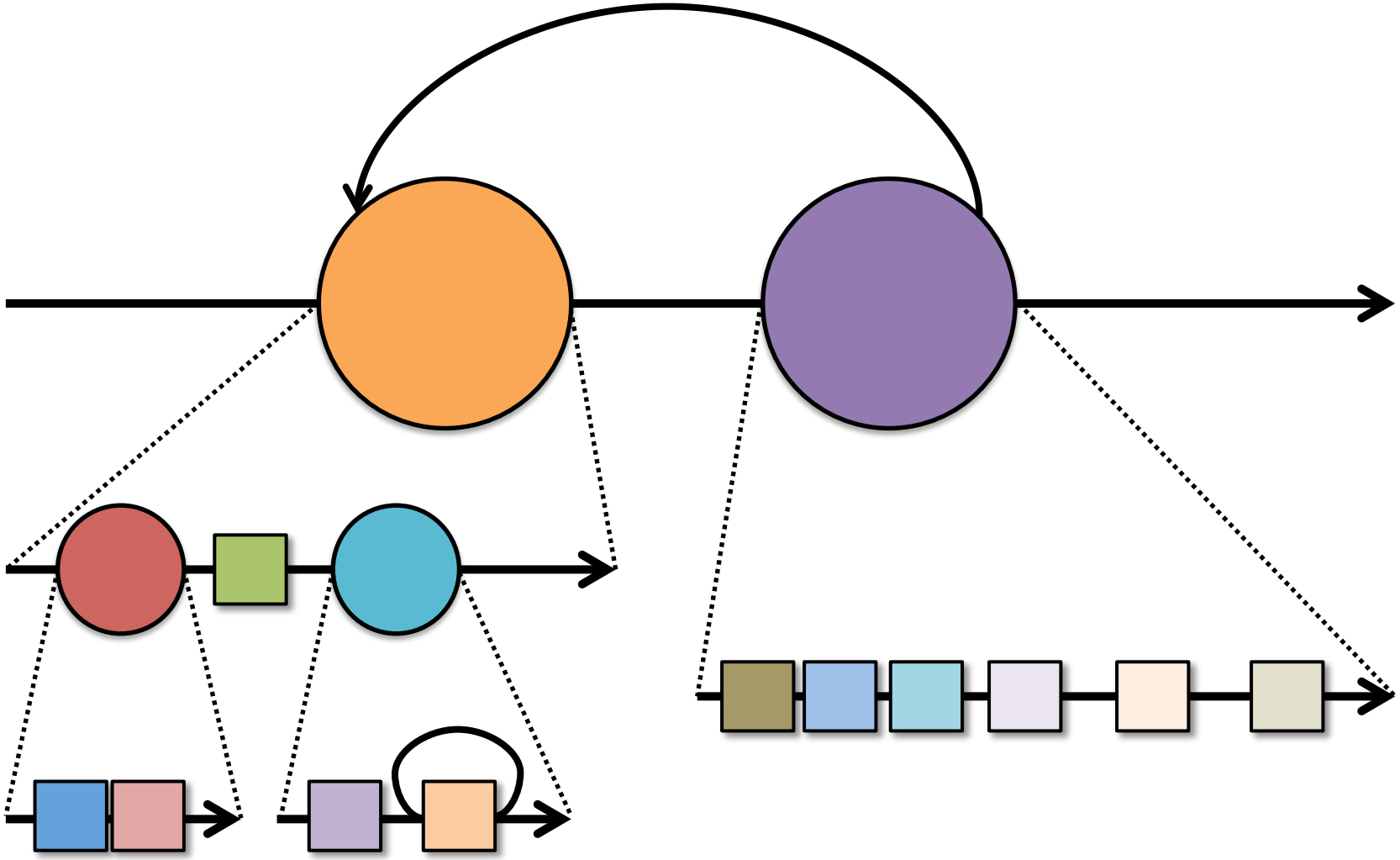






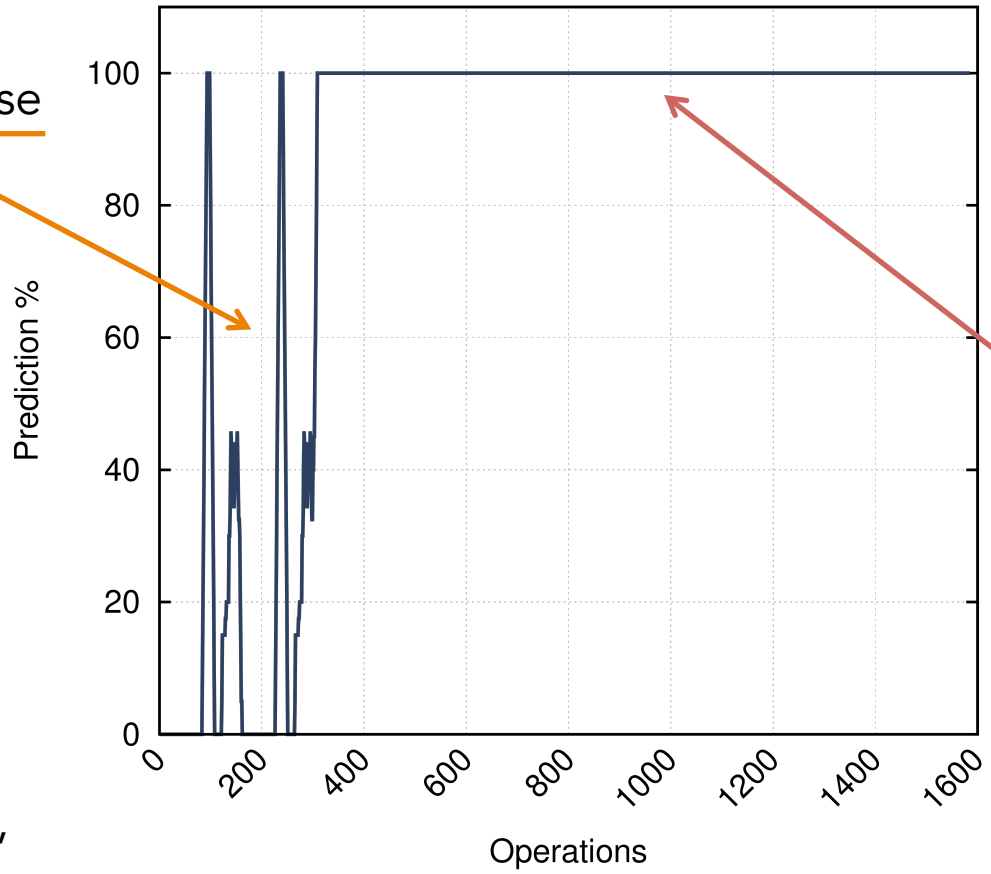


Build Rules, Build Models



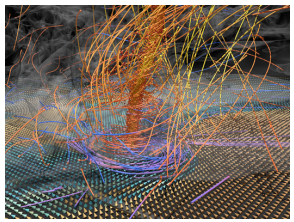
Example: Grammar Models with Omnisc'IO

Learning Phase



Fully Predictable Behavior

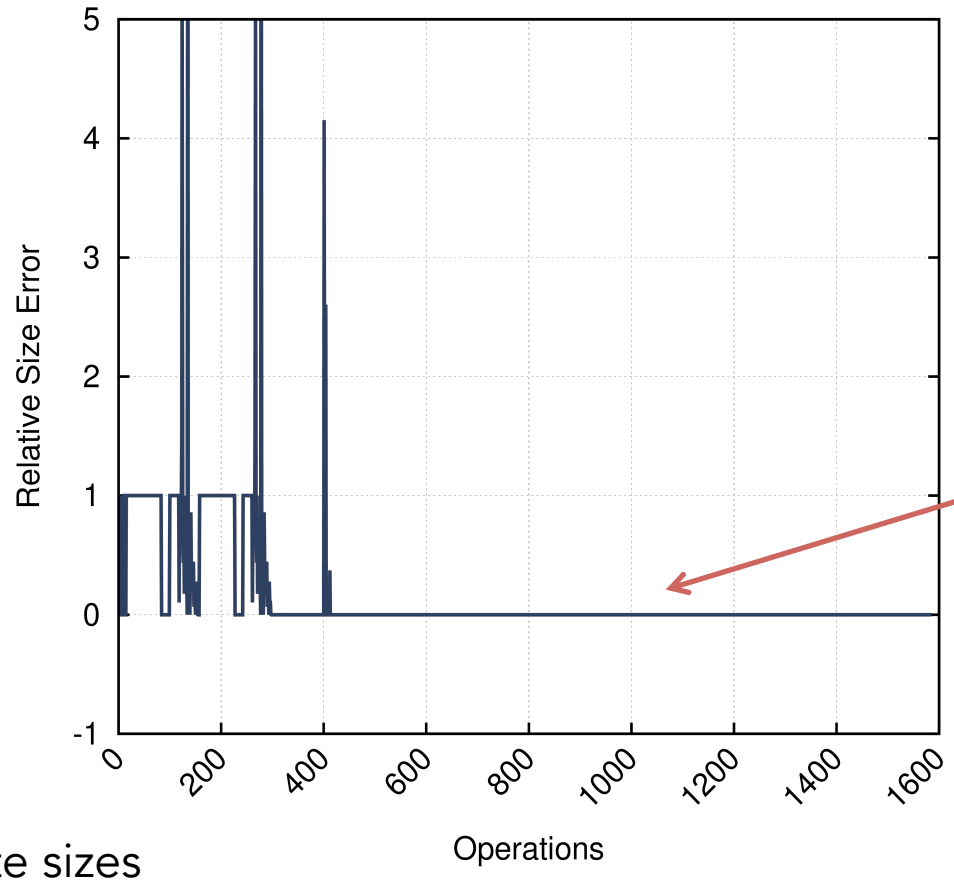
CM1



237 "colors"

M. Dorier, S. Ibrahim, G. Antoniu, R. Ross. *Using Formal Grammars to Predict I/O Behaviors in HPC: the Omnisc'IO Approach*. IEEE Transactions on Parallel and Distributed Systems (TPDS)

Predicting the size of future accesses

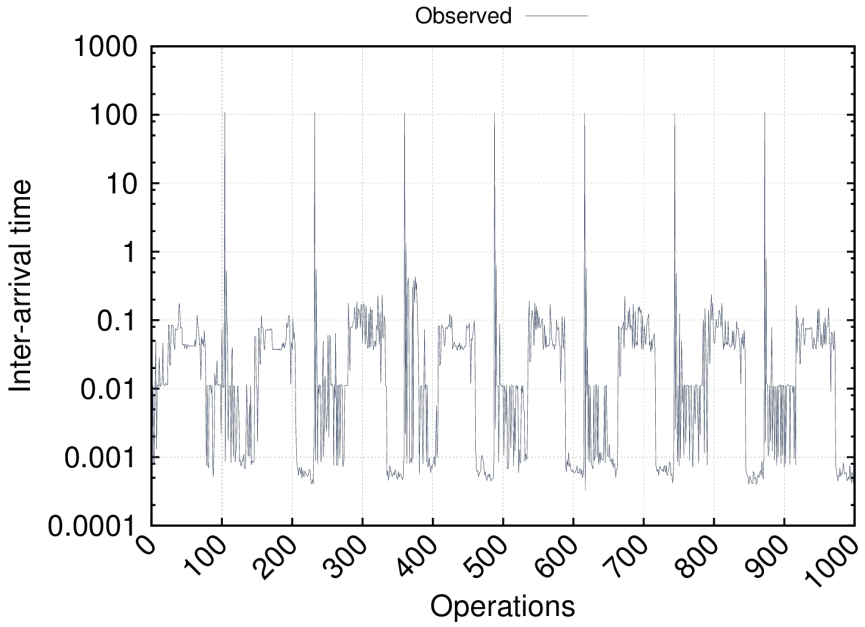


Fully Predictable
Access sizes

32 different write sizes
ranging from 3 B to 2.2 MB

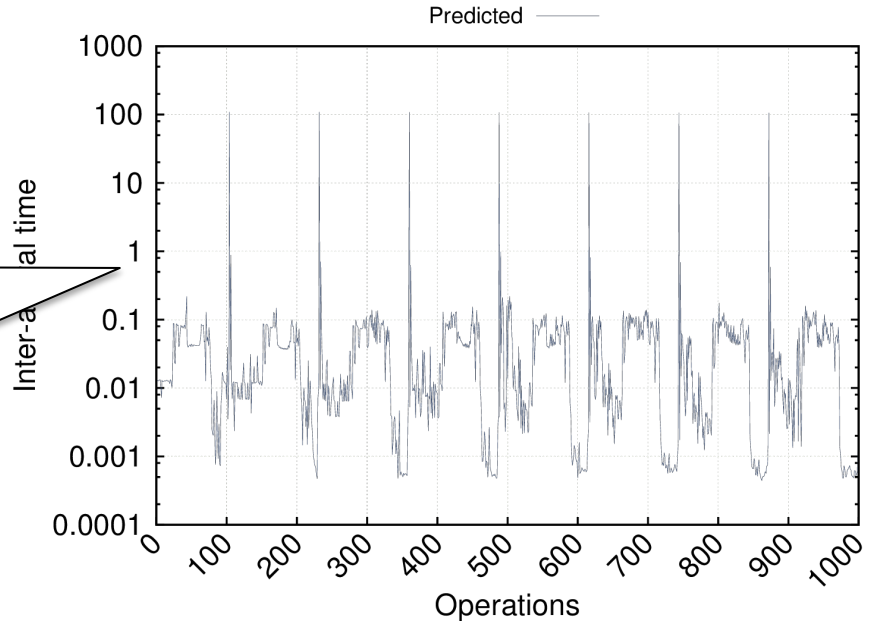
M. Dorier, S. Ibrahim, G. Antoniu, R. Ross. *Using Formal Grammars to Predict I/O Behaviors in HPC: the Omnisc'IO Approach*. IEEE Transactions on Parallel and Distributed Systems (TPDS)

Predicting the date of future accesses



Seasonal data, time series, let's use an ARIMAL model!

Just grammar and simple averages!



What can the machine do with that?

- Solve I/O Interference:
  - Better scheduling of I/O requests
- Better resource management
  - Burst Buffers, I/O Servers
  - Caching, Read-Ahead
- Simulation of future I/O behaviors
- Better trace compression…

Take away

Find determinism first
Use randomness if necessary

Use the predictability of I/O

Thank you!