Efficient Data Exchange in Amplpy via Apache Arrow Integration

Jürgen Lentz | lentz@ampl.com

2025 INFORMS Annual Meeting | Atlanta, Georgia
October 28 | 2:45 pm - 4:00 pm
Software for Optimization Modeling and Deployment 1

Outline

- Motivation
- Apache Arrow
- Benchmarks of amplpy
- Future development on amplpy

Motivation

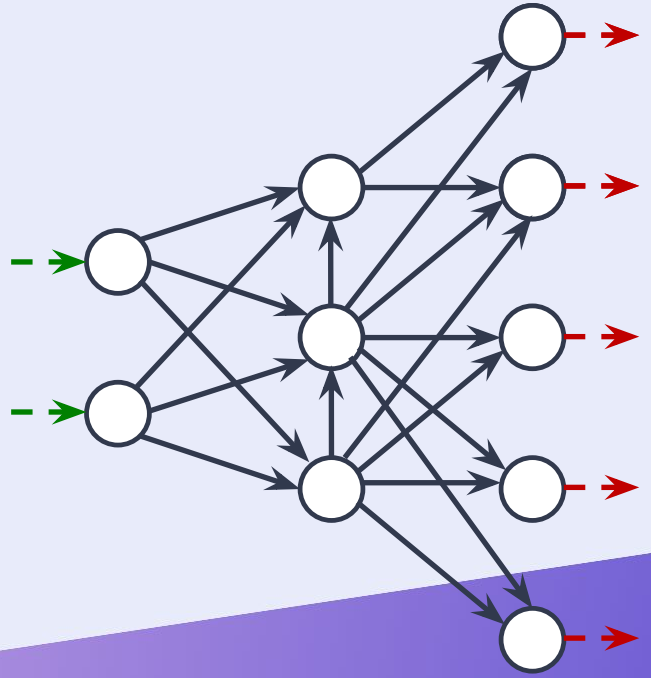
Example: Multi-Product Network Flow

Motivation

- Ship products efficiently to meet demands

Data

- a transportation network
  - Nodes representing cities
  - arcs representing roads
- supplies at nodes
- demands at nodes
- capacities on arcs
- shipping costs on arcs



AMPL Model for Multi-Product Network Flow

```
set PRODUCTS;
set NODES;
param net_inflow {PRODUCTS,NODES};

set ARCS within {NODES,NODES};
param capacity {ARCS} >= 0;

param var_cost {PRODUCTS,ARCS} >= 0;


var Flow {PRODUCTS,ARCS} >= 0;

minimize TotalCost:
   sum {p in PRODUCTS, (i,j) in ARCS} var_cost[p,i,j] * Flow[p,i,j];

subject to Capacity {(i,j) in ARCS}:
   sum {p in PRODUCTS} Flow[p,i,j] <= capacity[i,j];

subject to Conservation {p in PRODUCTS, j in NODES}:
   sum {(i,j) in ARCS} Flow[p,i,j] + net_inflow[p,j] =
   sum {(j,i) in ARCS} Flow[p,j,i];
```


AMPL Data for Multi-Product Network Flow

```
ampl = AMPL()


ampl.read("model_path")


ampl.set['PRODUCTS'] = products_series
ampl.set['NODES'] = nodes_series
ampl.param['net_inflow'] = net_inflow_df
ampl.set['ARCS'] = arcs_df
ampl.param['capacity'] = capacity_df
ampl.param['var_cost'] = var_cost_df
```


Runtime Comparison for Multi-Product Network Flow

```
ampl = AMPL()


ampl.read("model_path")


ampl.set['PRODUCTS'] = products_series
ampl.set['NODES'] = nodes_series
ampl.param['net_inflow'] = net_inflow_df
ampl.set['ARCS'] = arcs_df
ampl.param['capacity'] = capacity_df
ampl.param['var_cost'] = var_cost_df
```

Products = 100 NODES = 400 ARCS = 160,000

→ 0.000109 s

→ 49.050584 s

Apache Arrow

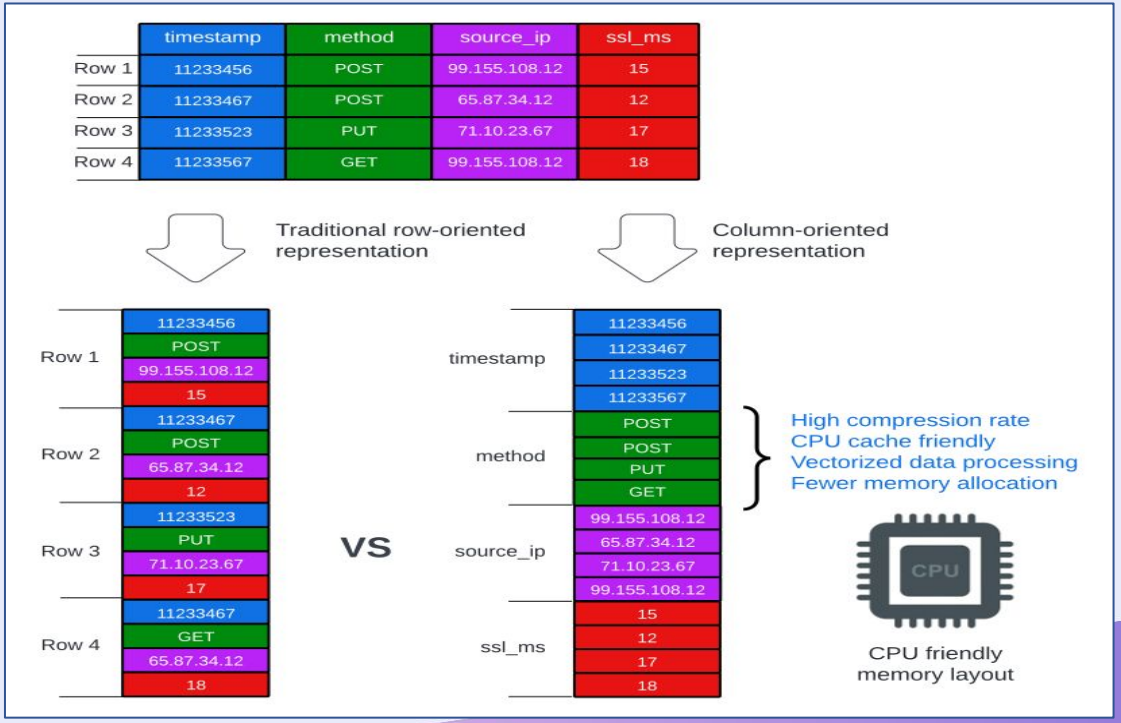
Apache Arrow

- Software framework for high-performance data processing and transport
- Key features and benefits of Apache Arrow

1. Columnar Memory Format
2. Language-Agnostics
3. Zero-Copy Reads
4. Integration with Data Ecosystem

Columnar Memory Format

Columnar Memory Format



Language-Agnostic

Language-Agnostic

Language-Agnostic

Language-Agnostic

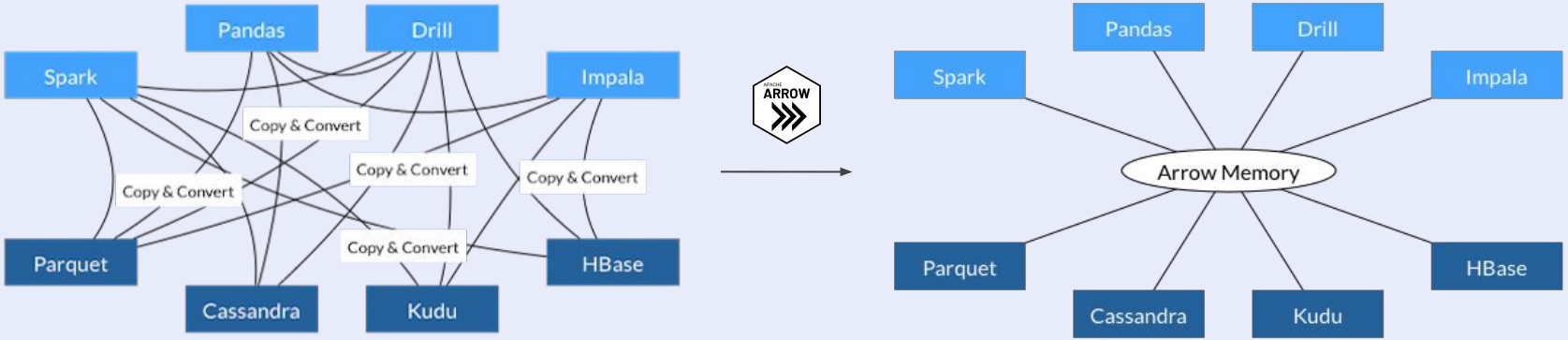
- Crossing the Language Barrier with the Arrow C Data API
    - Languages communicate via foreign function interface
    - Stable ABI for zero-copy sharing of data
    - Small-sized libraries

Zero-Copy Reads

Zero-Copy Reads

- Direct access to the data without the need to copy or move it
    - Lightning-fast data access
    - Avoiding the serialization overhead

Zero-Copy Reads: Standardization



Integration with Data-Ecosystem

Integration with Data Ecosystem

- Data Science and Analytics Tools

- Big Data Frameworks

- Databases and Storage Systems

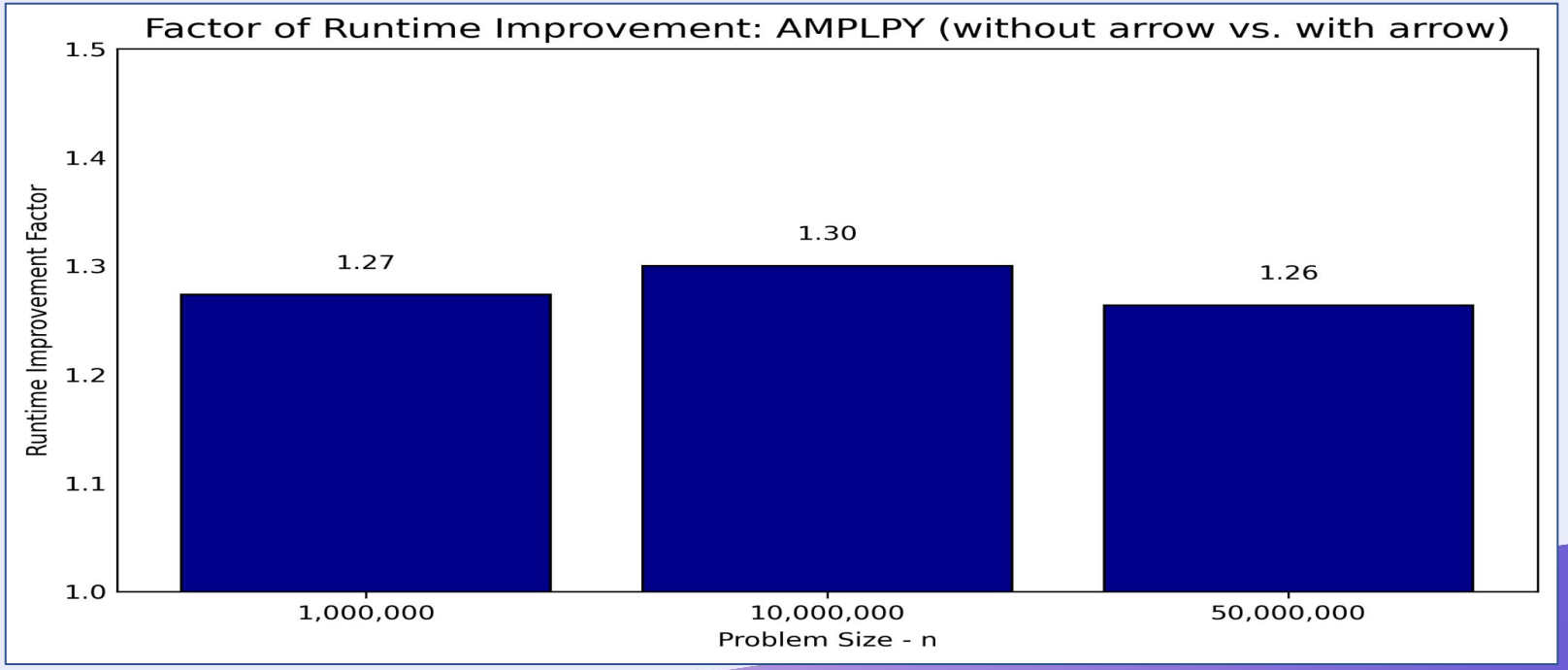
- Machine Learning

Benchmark of amplpy

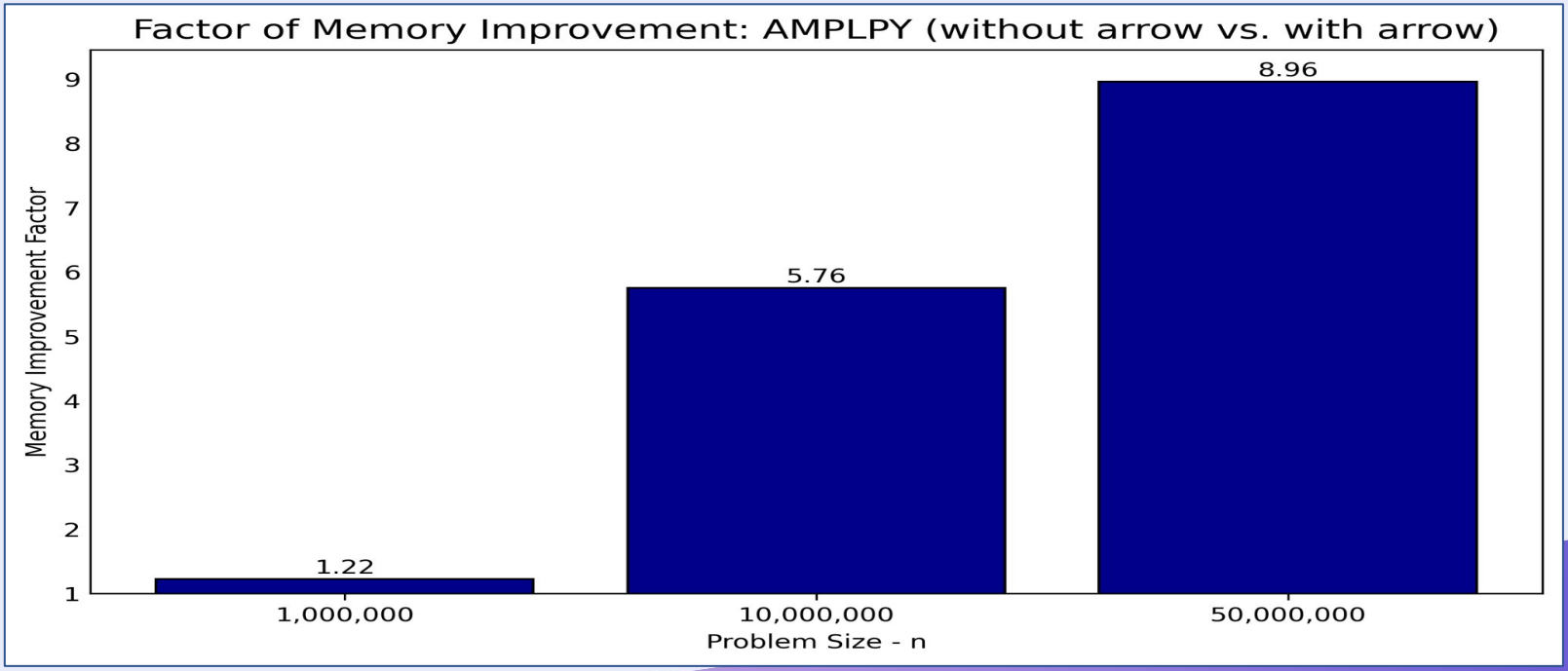
Benchmark Code - Integers

```
import pandas as pd
from amplpy import AMPL

n = 50_000_000
df = pd.DataFrame({"p": [i * 10 for i in range(1, n+1)]})
df.index = range(1, n+1)

ampl = AMPL()
ampl.eval("param n; param p{1..n};")
ampl.param['n'] = n
ampl.param['p'] = df
```

Runtime



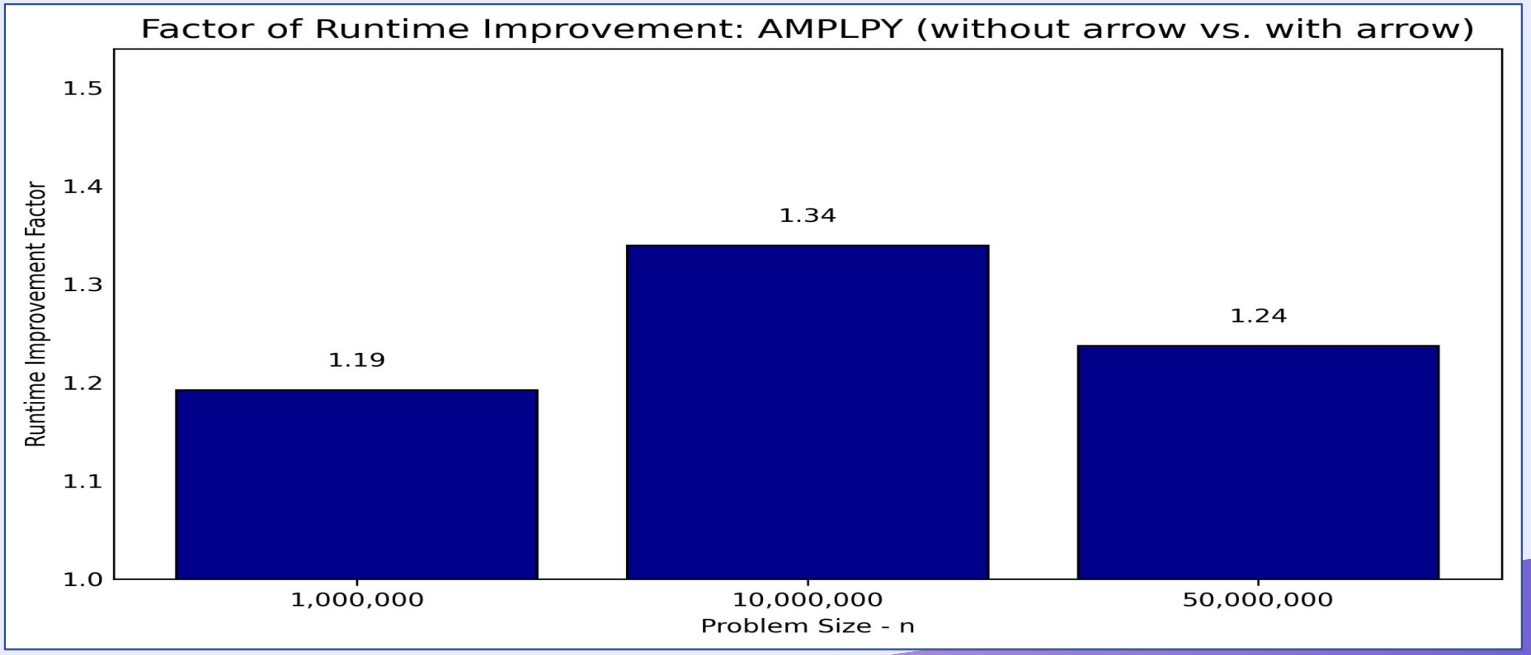
Memory



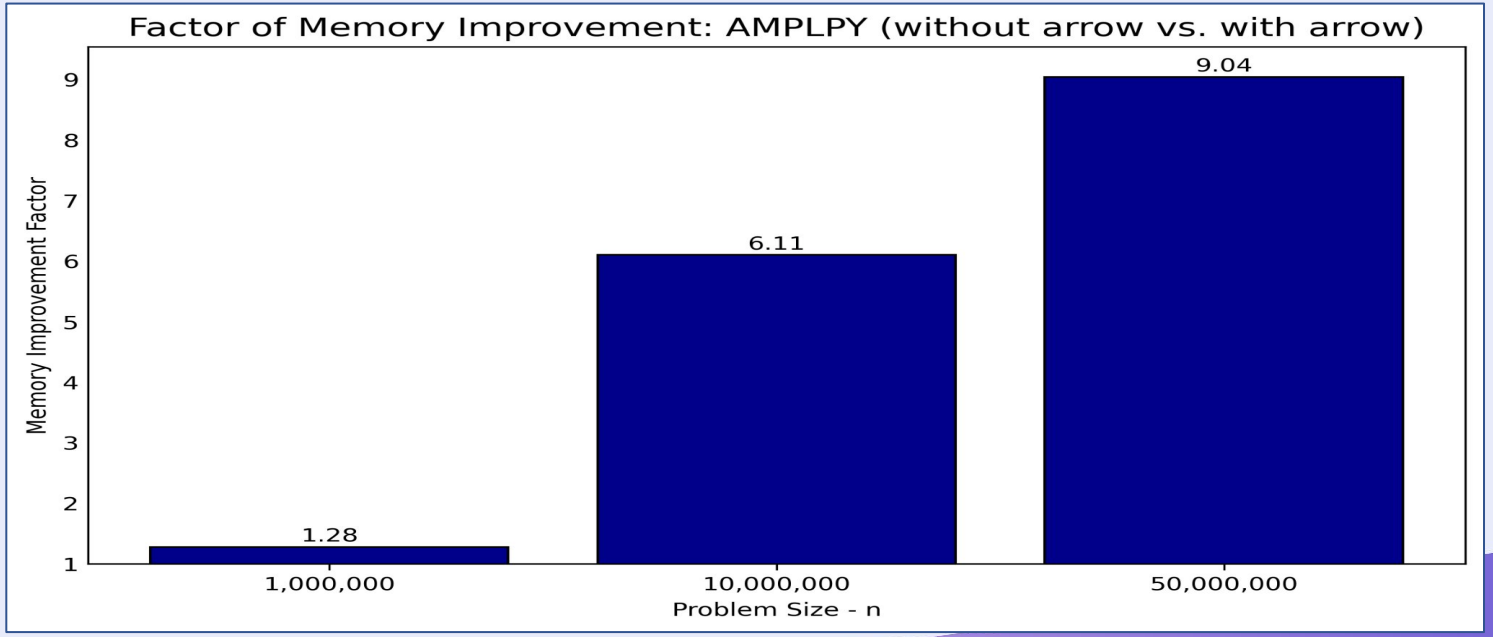
Benchmark Code - Floats

```
import pandas as pd
from amplpy import AMPL

n = 50_000_000
df = pd.DataFrame({"p": [float(i * 10.5) for i in range(1, n+1)]})
df.index = range(1, n+1)

ampl = AMPL()
ampl.eval("param n; param p{1..n};")
ampl.param['n'] = n
ampl.param['p'] = df
```


Runtime



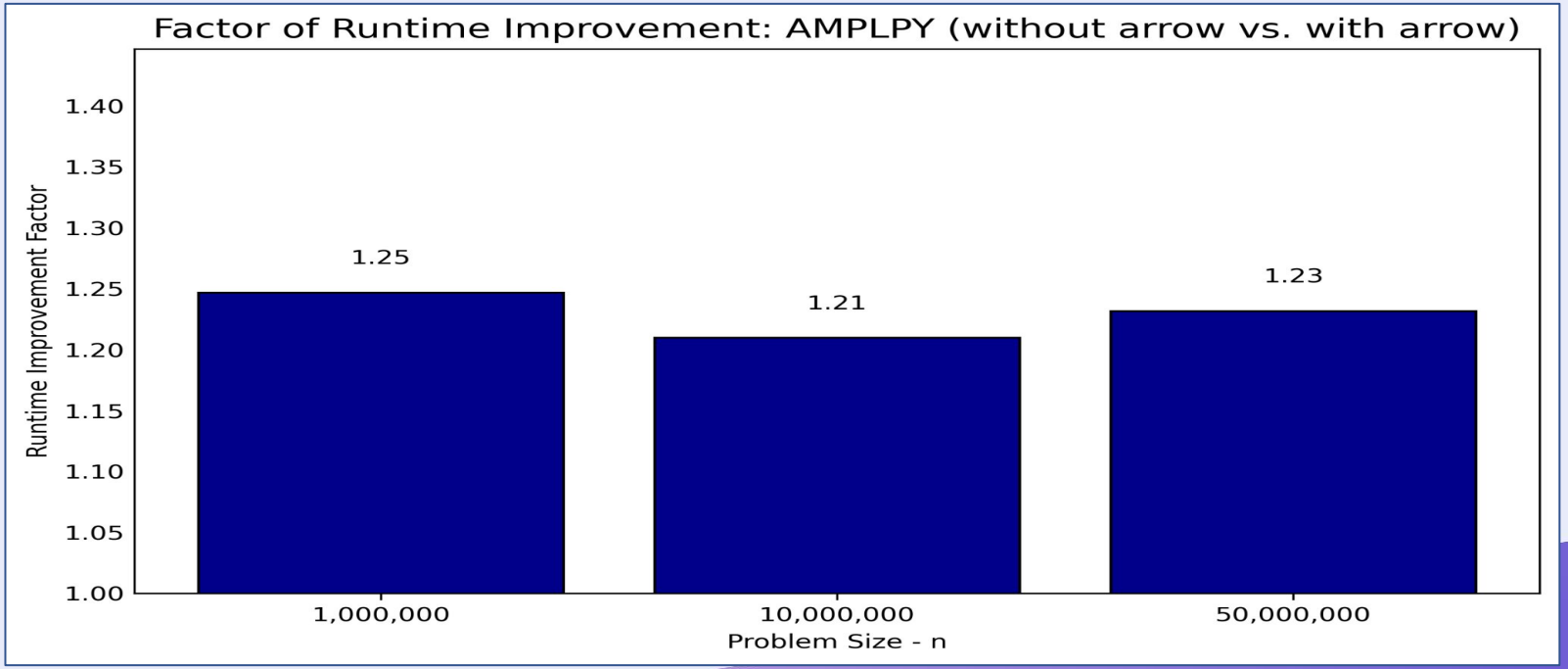
Memory



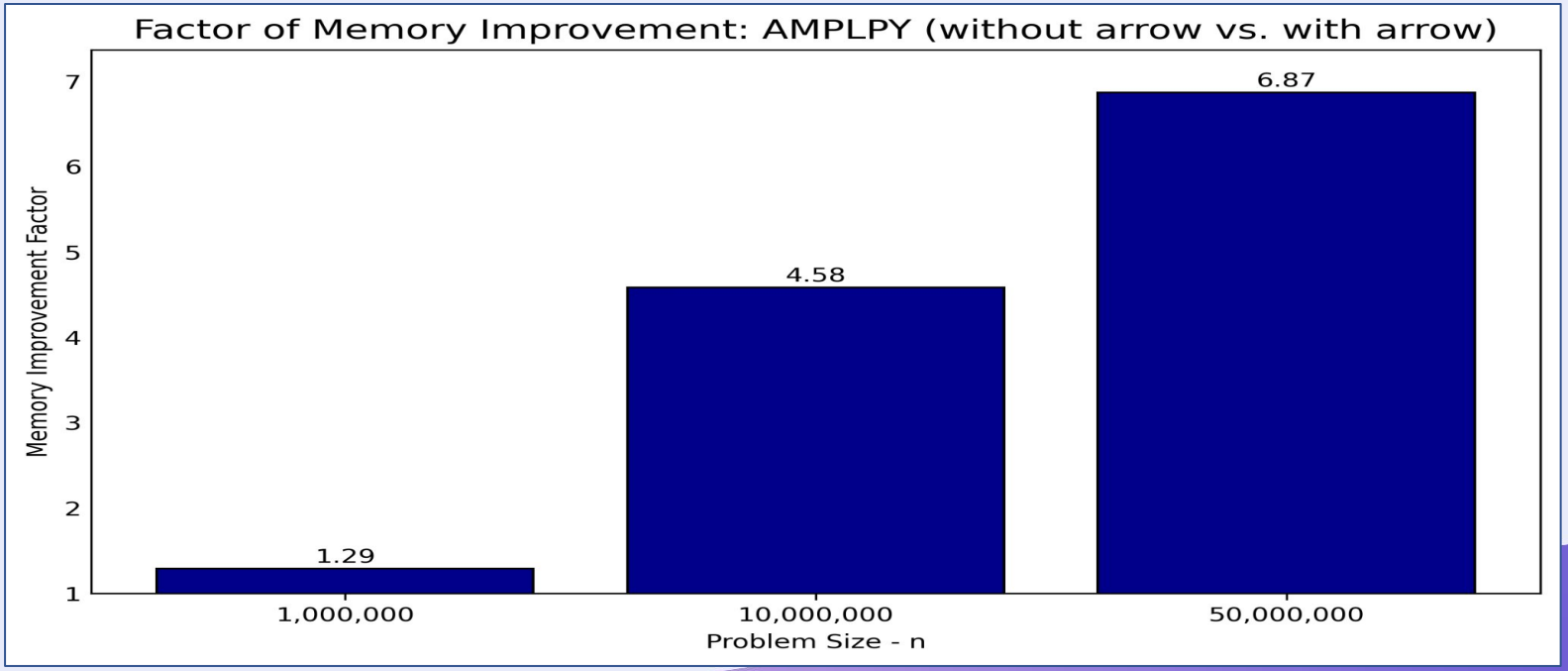
Benchmark Code - Strings

```
import pandas as pd
from amplpy import AMPL
import random
import nltk

nltk.download("words")
word_list = nltk.corpus.words.words()
n = 50_000_000
random.seed(n)
df = pd.DataFrame({"p": [random.choice(word_list) for _ in range(1, n+1)]})
df.index = range(1, n+1)
ampl = AMPL()
ampl.eval("param n; param p{1..n} symbolic;")
ampl.param['n'] = n
ampl.param['p'] = df
```


Runtime



Memory


Factor of Memory Improvement: AMPLPY (without arrow vs. with arrow)
Memory Improvement Factor
7
6
5
4
3
2
1
1.29
4.58
6.87
1,000,000
10,000,000
50,000,000
Problem Size - n

Future Development on amplpy

Future Development on amplpy

- Support more packages from the Data Ecosystem
- Show and expand functionalities of ampl
- Support for type hints
- amplpy no-GIL packages

Learn more

amplpy.ampl.com

→ Documentation of amplpy

amplpy.ampl.com/en/latest/reference.html

→ API reference of amplpy

github.com/ampl/amplpy

→ Github repository of amplpy

ampl.com/mo-book

→ New AMPL+Python book

ampl.com/colab

→ Collection of AMPL models in Jupyter Notebooks

