# Predicting the future!

## And scaling a streaming system based on it

Frank Verkoren (fverkoren@tudelft.nl) | Supervisors: Asterios Katsifodimos, Oto Mráz

TU Delft Delft University of Technology

### 1. Predicting the Future (Introduction)

Imagine being able to know the future and know exactly how much data will be flowing through your system (throughput).

- Reactive scaling is late and causes lag.
- **Overprovisioning** wastes resources to prevent lag.
- Predictive scaling prevents lag. We propose forecasting.
- We test this using **Styx**, a distributed streaming system (Stateful FaaS) whose lack of native autotuning makes it the perfect testbed.

### 2. The Challenge (Research Question)

*How can a time series forecasting policy for automatic resource allocation be designed and integrated into an autotuning system?*

### 3. What We Tried (Background & Models)

Four forecasting models were analysed offline to find the best configurations for each model:

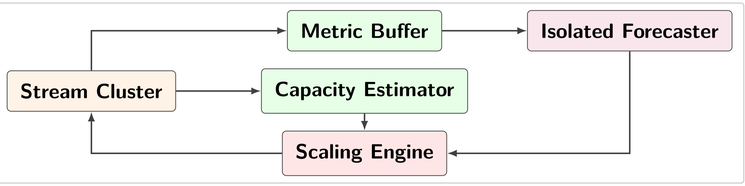
- **Chronos (LLM):** Treats time-series as natural language. Highly accurate but computationally expensive.
- **River (Statistical):** An online streaming SNARIMAX model. Fast, continuous online forecasting.
- **LSTM & GRU (RNNs):** Recurrent Neural Networks good at approximating nonlinear trends.

### 4. How It Works (Method & Implementation into Styx)

We integrated the optimal offline configurations into Styx, creating a flexible framework that continuously watches incoming throughput and predicts future demand.

- Measure current load (how much data enters the system).
- Predict future load (how much data will enter the system).
- Decide how many workers are needed.
- Add workers before spikes happen.
- Remove workers when demand drops.

**Figure 1: Predictive Scaling Architecture**




### 5. Did It Work? (Results)

Yes. Exact numbers are shown in Table 1.

**Table 1: Summary Statistics Across Configurations**

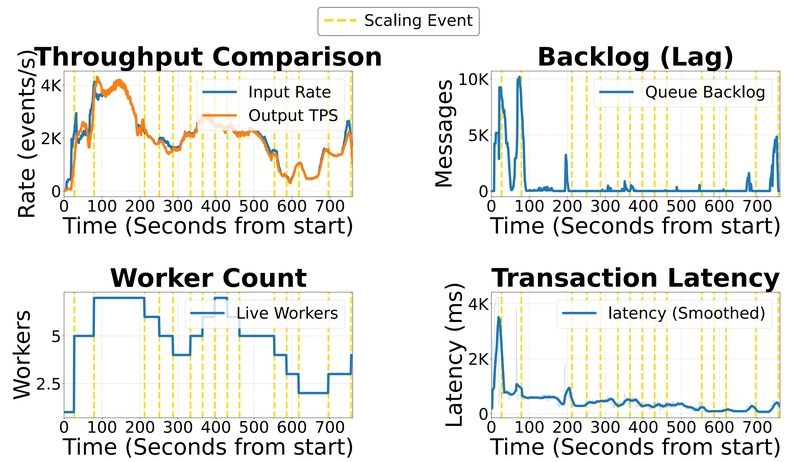
| Configuration | Workers | Avg TPS | Avg Backlog | p50 Latency |
|---|---|---|---|---|
| **Chronos** | 4.77 | 2061 | **672** | **328ms** |
| River | 5.04 | 2162 | 7902 | 365ms |
| LSTM | 4.66 | 2105 | 1740 | 497ms |
| GRU | 4.51 | 2083 | 5434 | 576ms |
| *5 workers (Static)* | *5.00* | *2067* | *6987* | *409ms* |

Chronos performed best compared to a static system using a similar number of workers, Chronos achieved a **90% reduction** in average backlog, an **82% reduction** in maximum backlog, and a **20% improvement** in median latency.

### 6. Why Chronos Won (Discussion)

Chronos's performance is **best** because it accurately mirrors throughput (Figure 2), but its predictions are also the most **expensive** computationally (100ms inference time).

**Figure 2: CHRONOS Autoscaling Performance**




### 7. Conclusions & Future Work

- **Conclusion:** Predictive scaling mathematically outperforms static allocation systems under the same resource budget.
- **Future Work (Offline Training):** Pre-training forecasting models on historical data to drop latency overhead.
- **Future Work (Cold-Starts):** Retaining states of previously run models to prevent inaccurate scaling on bootup.

**Definitions**

**TPS**: Transactions Per Second • **FaaS**: Function-as-a-Service • **LLM**: Large Language Model • **RNN**: Recurrent Neural Network
**LSTM**: Long Short-Term Memory • **GRU**: Gated Recurrent Unit • **SNARIMAX**: Seasonal Auto-Regressive Integrated Moving Average with eXogenous variables