ACHIEVING PERFORMANCE AND AVAILABILITY GUARANTEES WITH AMAZON SPOT INSTANCES

MICHELE MAZZUCCO*

UNIVERSITY OF TARTU

* And Marlon Dumas. Thanks to Madeline Gonzalez for the useful discussions.

INTRODUCTION (1)

- **Cloud computing = pay-as-you go + “unlimited” capacity**
- **Amazon Elastic Cloud Compute (EC2)**
  - Web service providing resizable compute capacity
- **Different instance types, e.g., small, large, xlarge servers**
- **Different purchasing options**
  1. On-Demand Instances
     - Pay for resources by the hour
     - No long-term commitments
  2. Reserved Instances
     - Upfront payment for reserving instances
     - Discounted usage rate

INTRODUCTION (2)

3. Spot Instances
    - Users bid for unused resources at a “limit price” (the maximum price the bidder is willing to pay)
    - Amazon gathers the bids and determines a clearing price (“spot price”) based on the bids and the available capacity
    - A bidder gets the required instances if his/her limit price is above the clearing price. In this case, the bidder pays the clearing price (not his/her limit price)
    - The clearing price is updated as new bids arrive. If the clearing price goes above the bidder’s limit price, the bidder’s running spot instances are terminated.

INTRODUCTION (3)

- **In this presentation I will focus on “spot instances”**
  - They allow IaaS <u>providers</u> to sell spare capacity (via auctions), thus improving resource utilization
  - At the same time they enable IaaS <u>users</u> to acquire resources at a price lower than on-demand price
    - However, spot instances are terminated any time the “clearing price” goes above the user’s “limit price”

PROBLEM STATEMENT

"Can we use spot instances to run paid web services while achieving performance and availability guarantees?"

TALK OUTLINE

Bidding strategy

- Decides how much to bid on the spot market

Meet availability goals

Server allocation

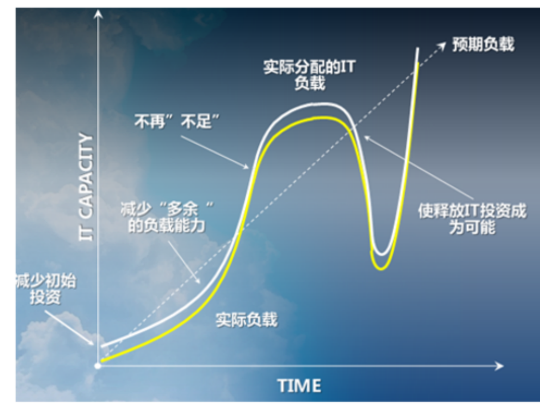
- Decides how many servers to use

Admission control

- Decides how many jobs to admit

Mission accomplished

Meet performance targets

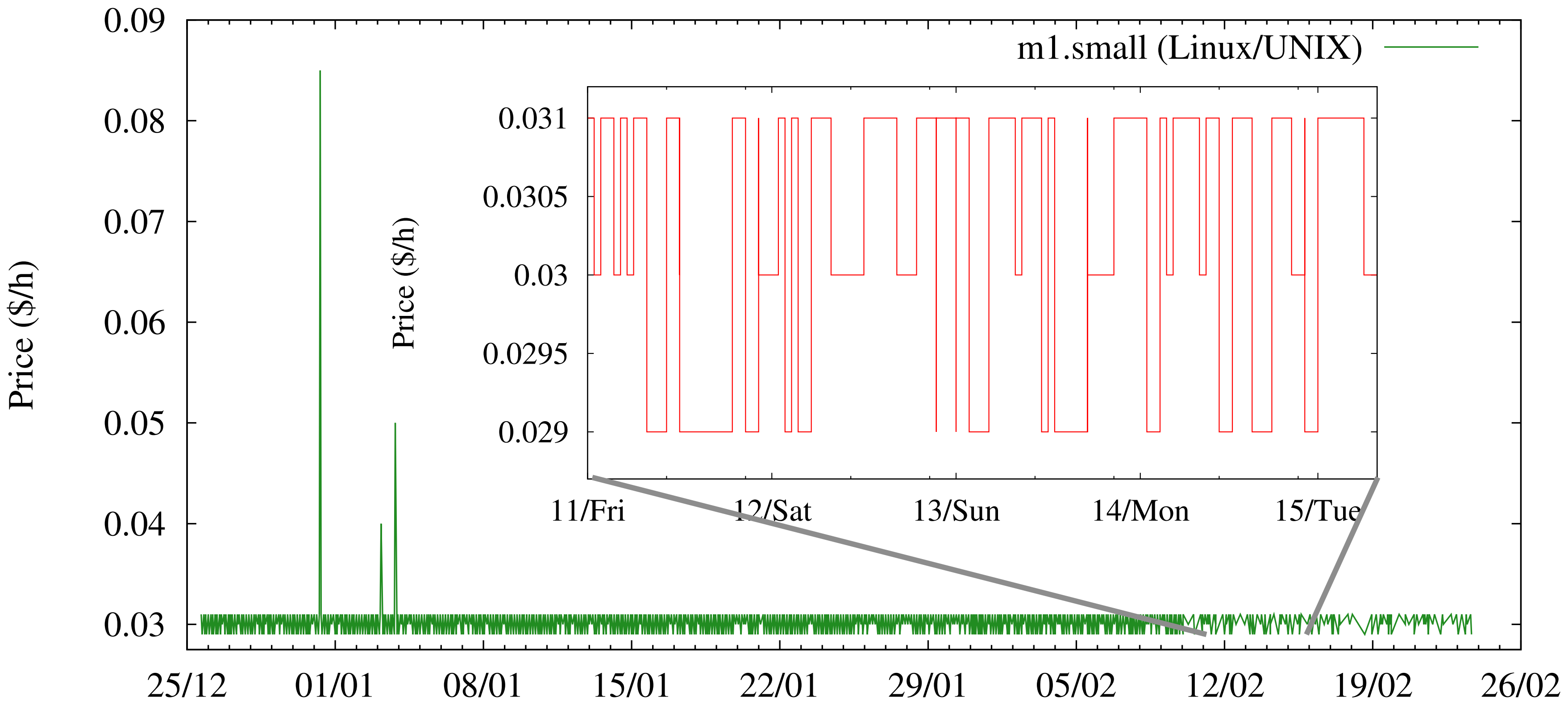


PART 1: SPOT PRICE PREDICTION

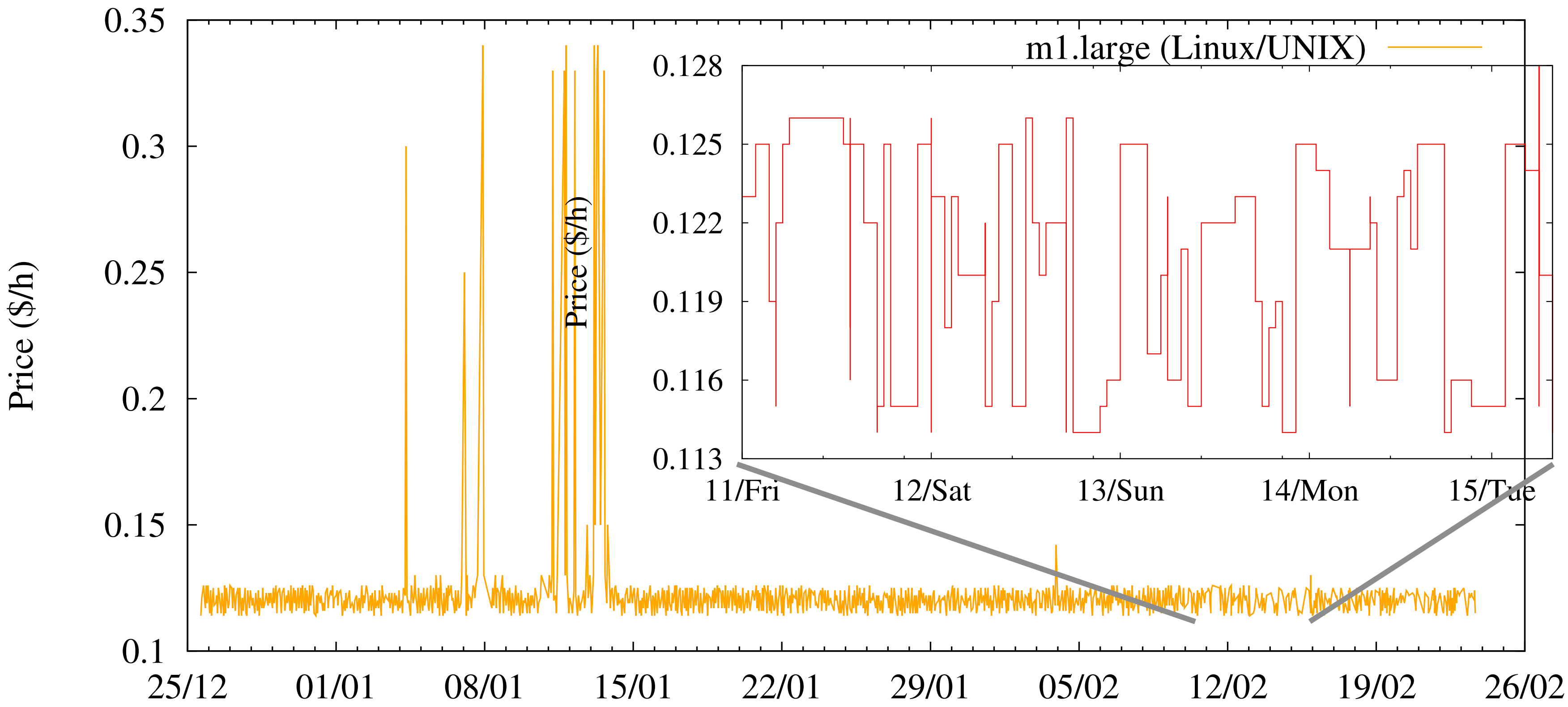
- **We must determine the optimal "limit price" that the SaaS provider should bid on the spot market**
  - The goal is to bid the lowest possible price capable of achieving the desired level of availability

- **Let's see how some spot prices look like…**

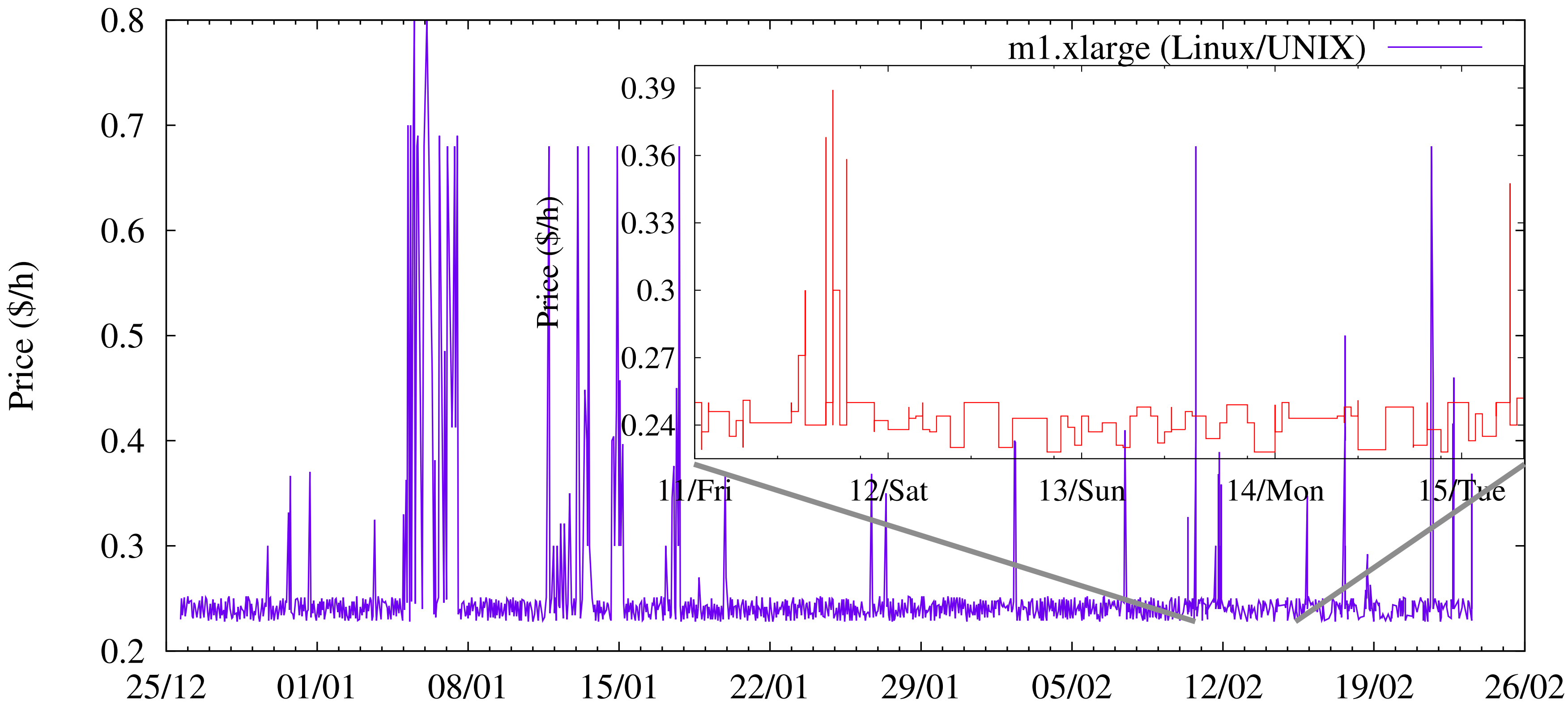
SPOT PRICES (M1.SMALL)



SPOT PRICES (M1.LARGE)



SPOT PRICES (M1.XLARGE)



1ST ATTEMPT

- **Assume that prices follow patterns, e.g., day/night, week days/week ends, etc.**
- **Use triple exponential smoothing ("Holt-Winters") for predicting future prices**

Constants minimizing the MSE

$$\begin{cases} S_t = \alpha \dfrac{y_t}{I_t} + (1-\alpha)(S_{t-1} + b_{t-1}) \\ b_t = \gamma(S_t - S_{t-1}) + (1-\gamma) b_{t-1} \\ I_t = \beta \dfrac{y_t}{S_t} + (1-\beta) I_{t-L} \end{cases}$$

$\longleftarrow$ Updates the smoothed value

$\longleftarrow$ Updates the trend

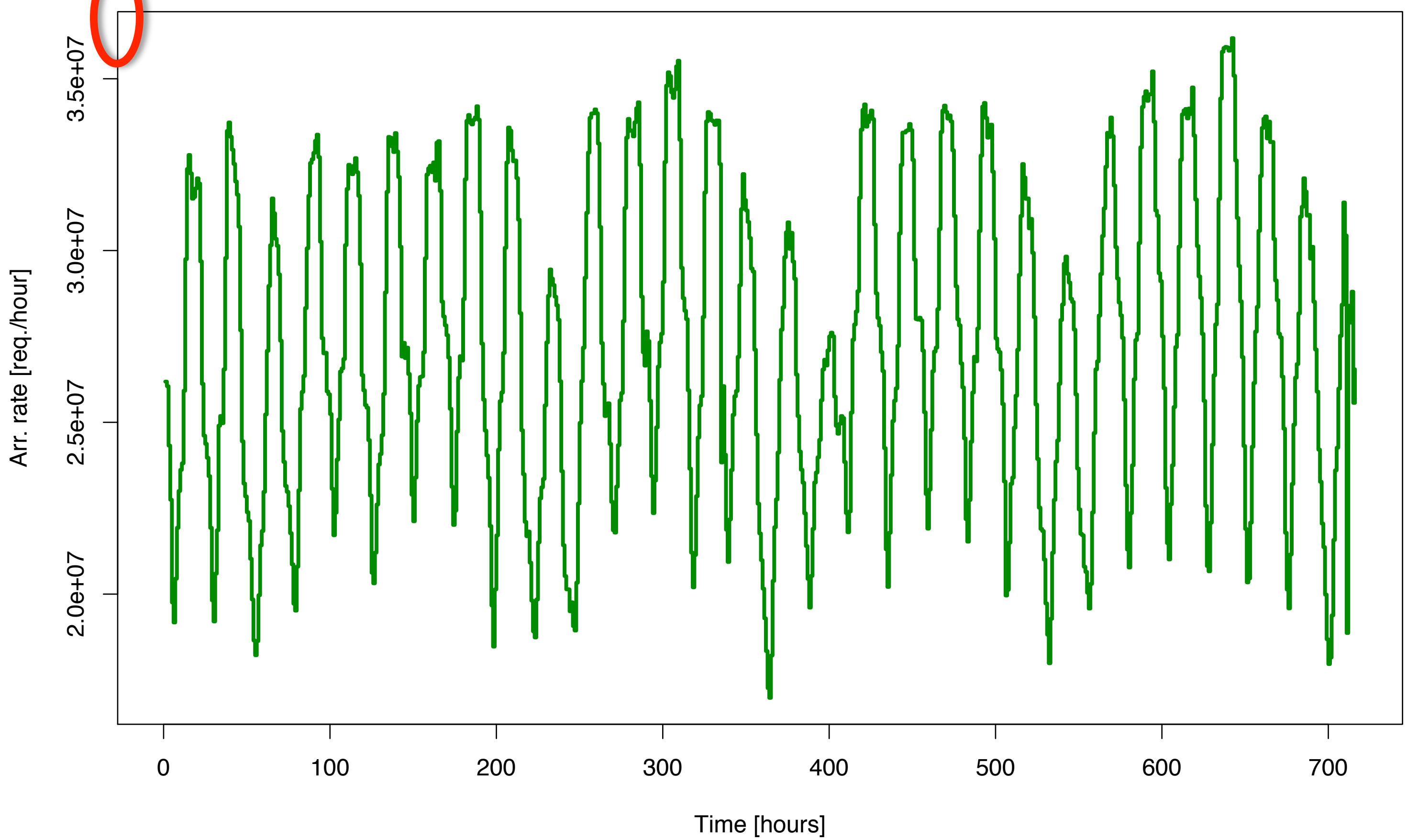
$\longleftarrow$ Updates the seasonal component

$$F_{t+m} = (S_t + m b_t) I_{t-L+m}$$

$\longleftarrow$ *m* periods ahead's forecast

WIKIPEDIA TRAFFIC

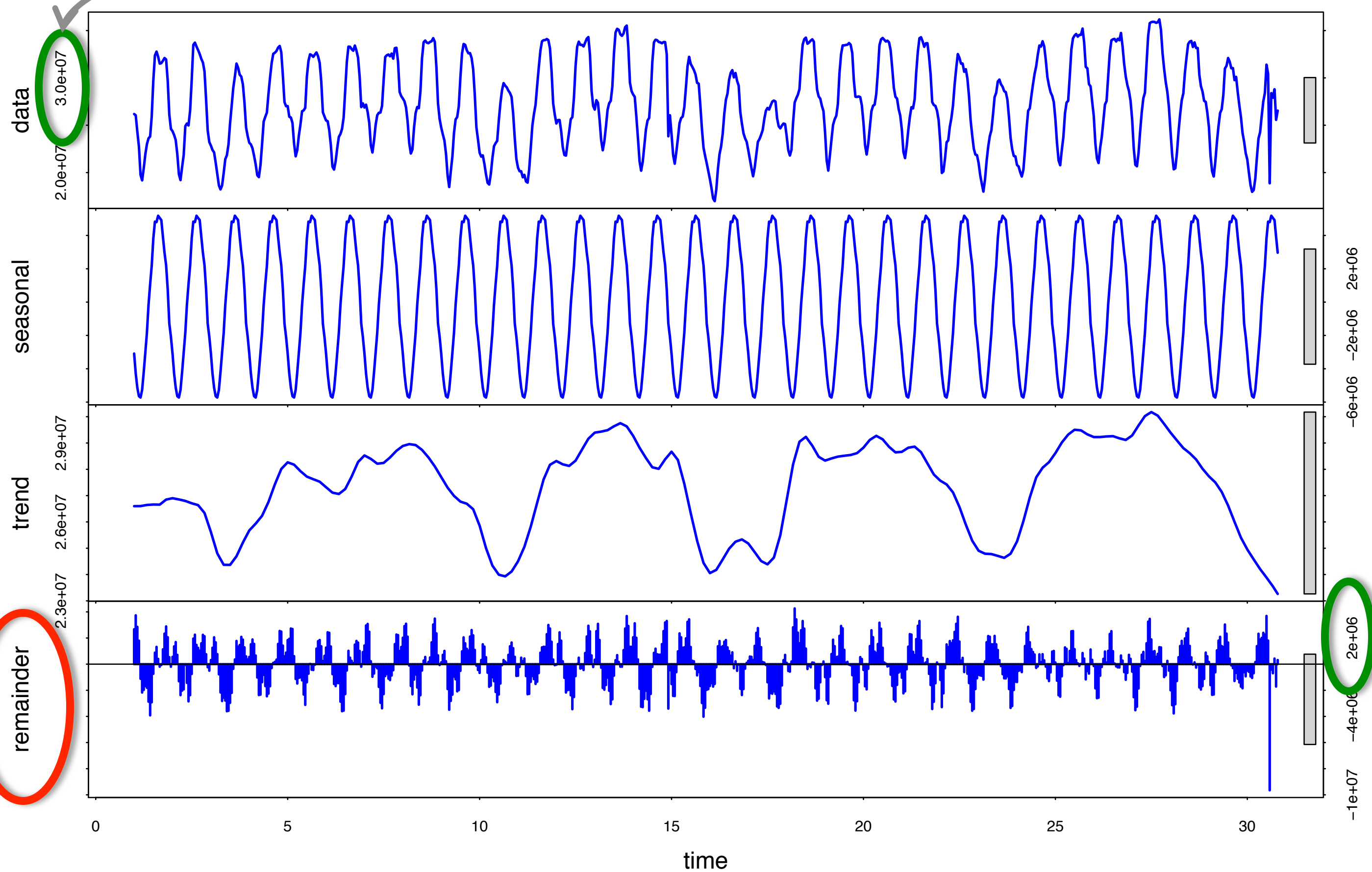
~ 10,000 req./sec at peak



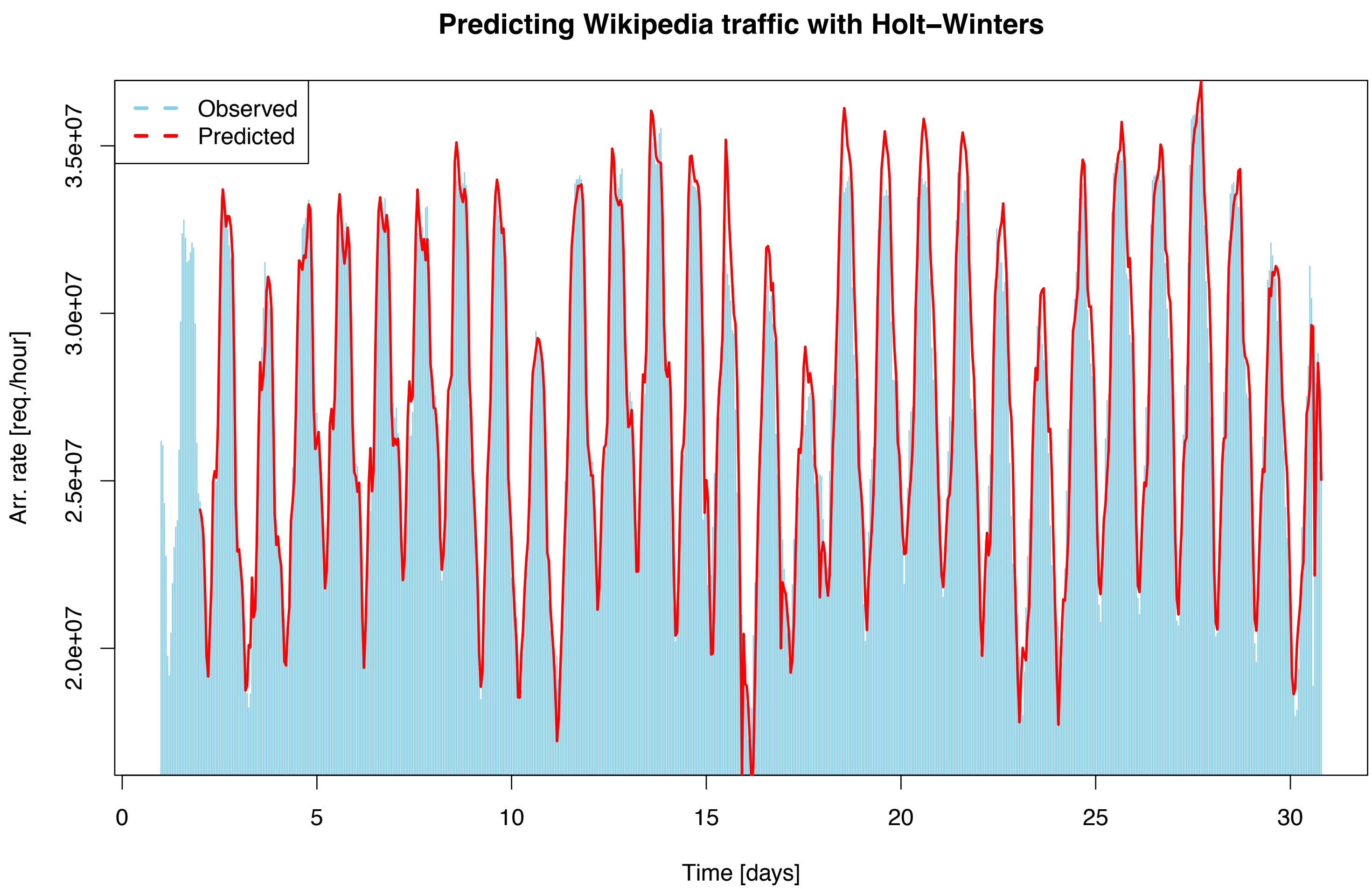
WIKIPEDIA TRAFFIC: TIME SERIES DECOMPOSITION

The max irregular component is ~11% of the data

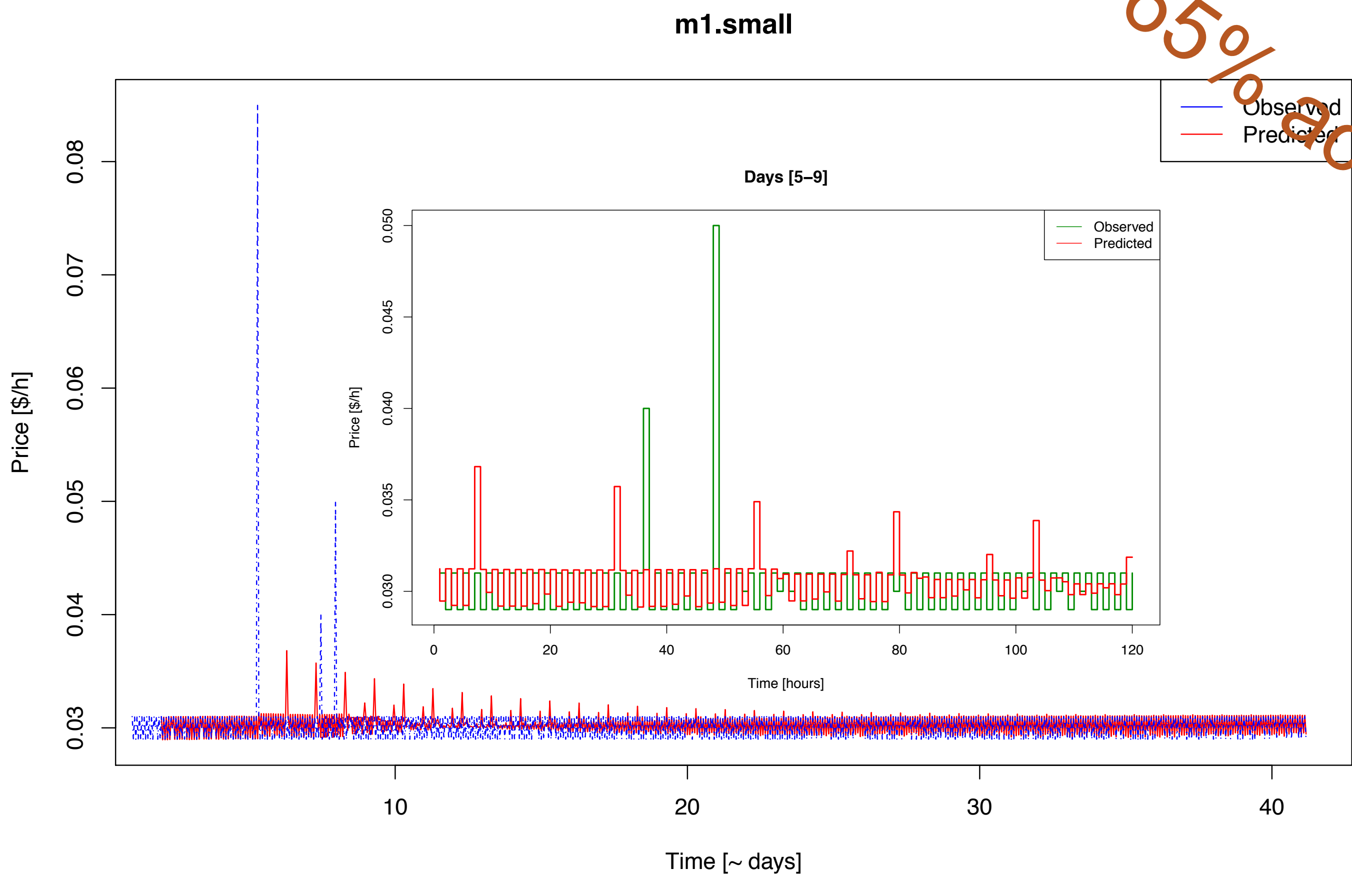
= data – (trend + seasonal)



PREDICTING WIKIPEDIA TRAFFIC WITH HOLT WINTERS

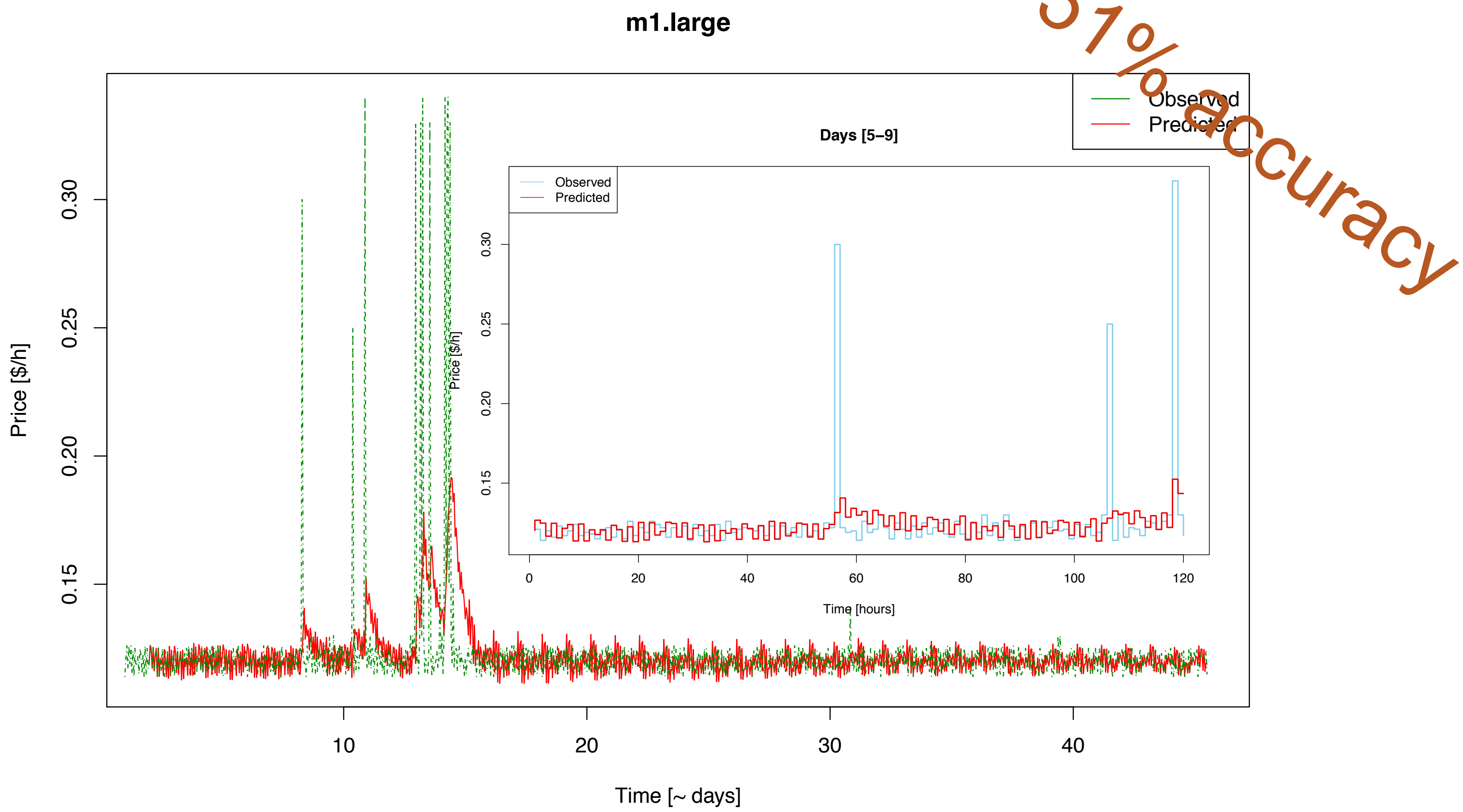


PREDICTING SPOT PRICES WITH HOLT WINTERS (1)

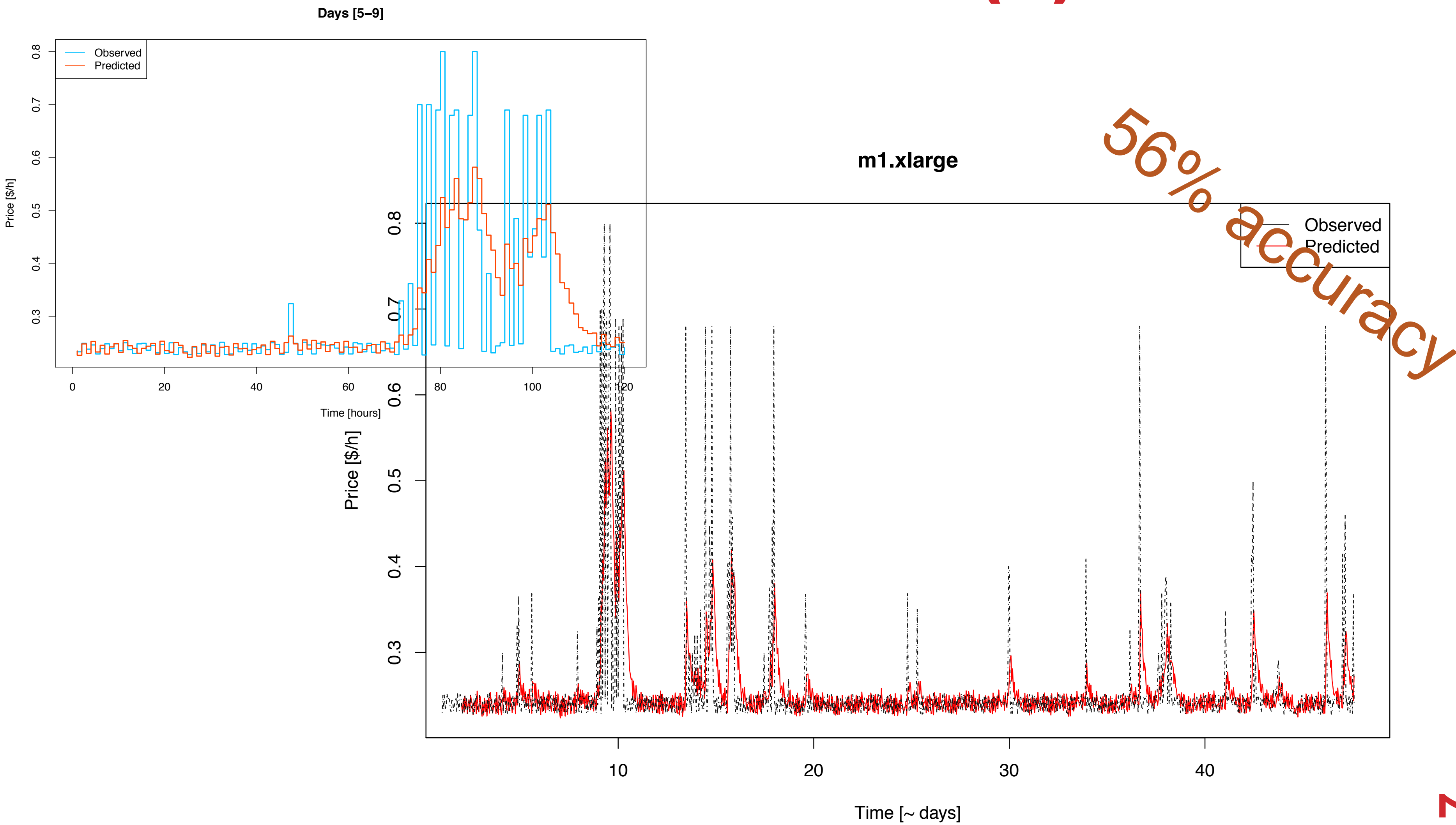


65% accuracy

PREDICTING SPOT PRICES WITH HOLT WINTERS (2)

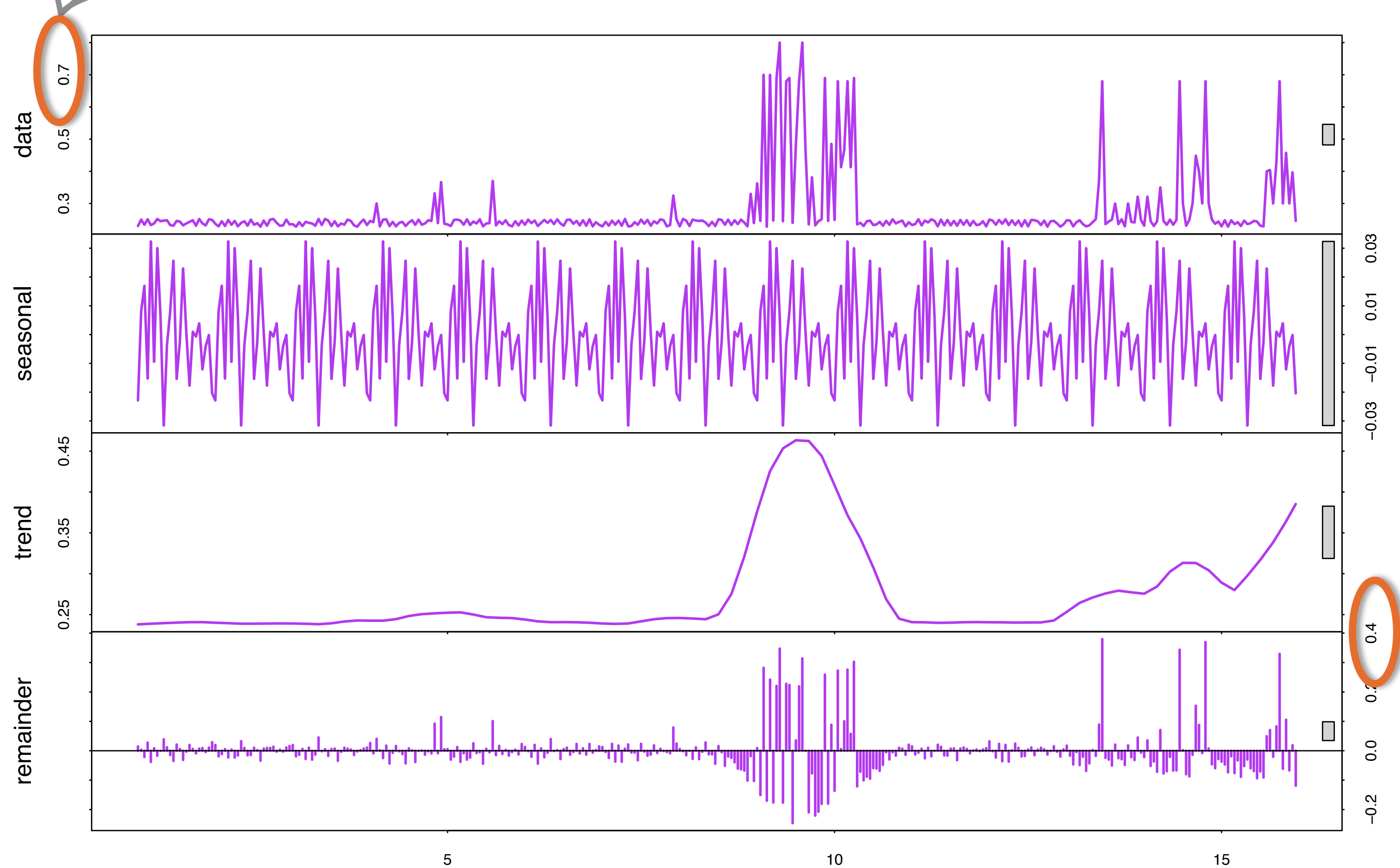


PREDICTING SPOT PRICES WITH HOLT WINTERS (3)



XLARGE PRICES: TIME SERIES DECOMPOSITION

The max irregular component is ~50% of the data

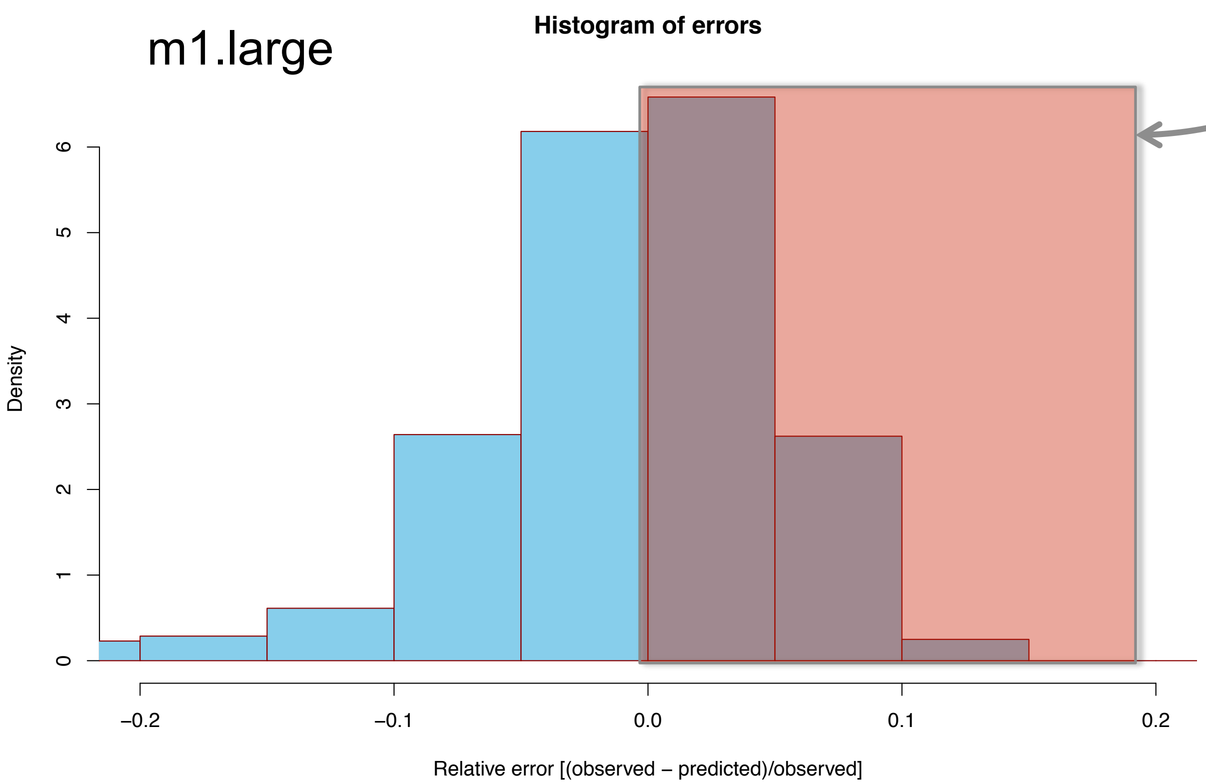


2ND ATTEMPT (1)

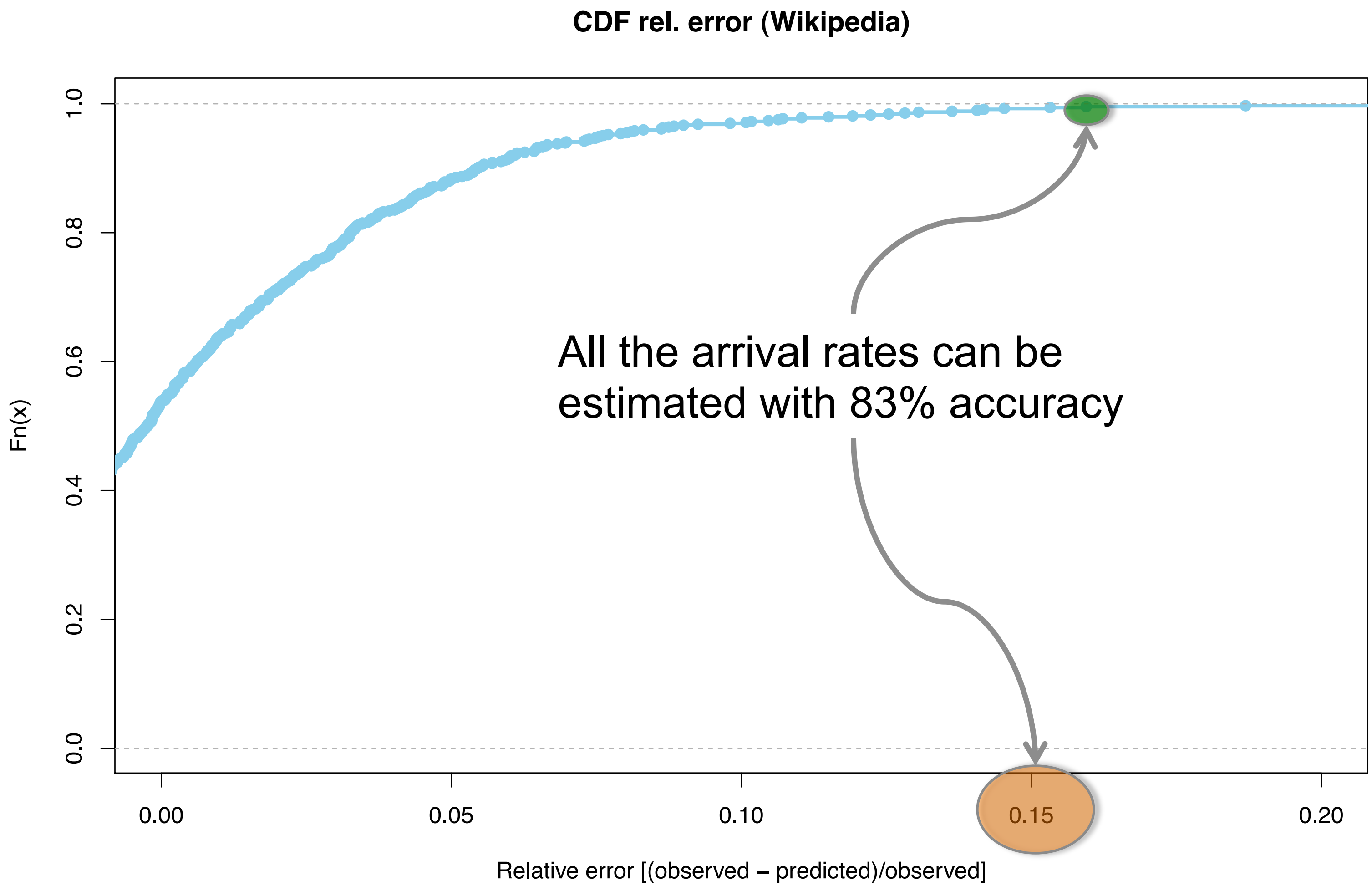
- **What about employing the distribution of the relative errors to correct the prediction?**

We are interested in this part

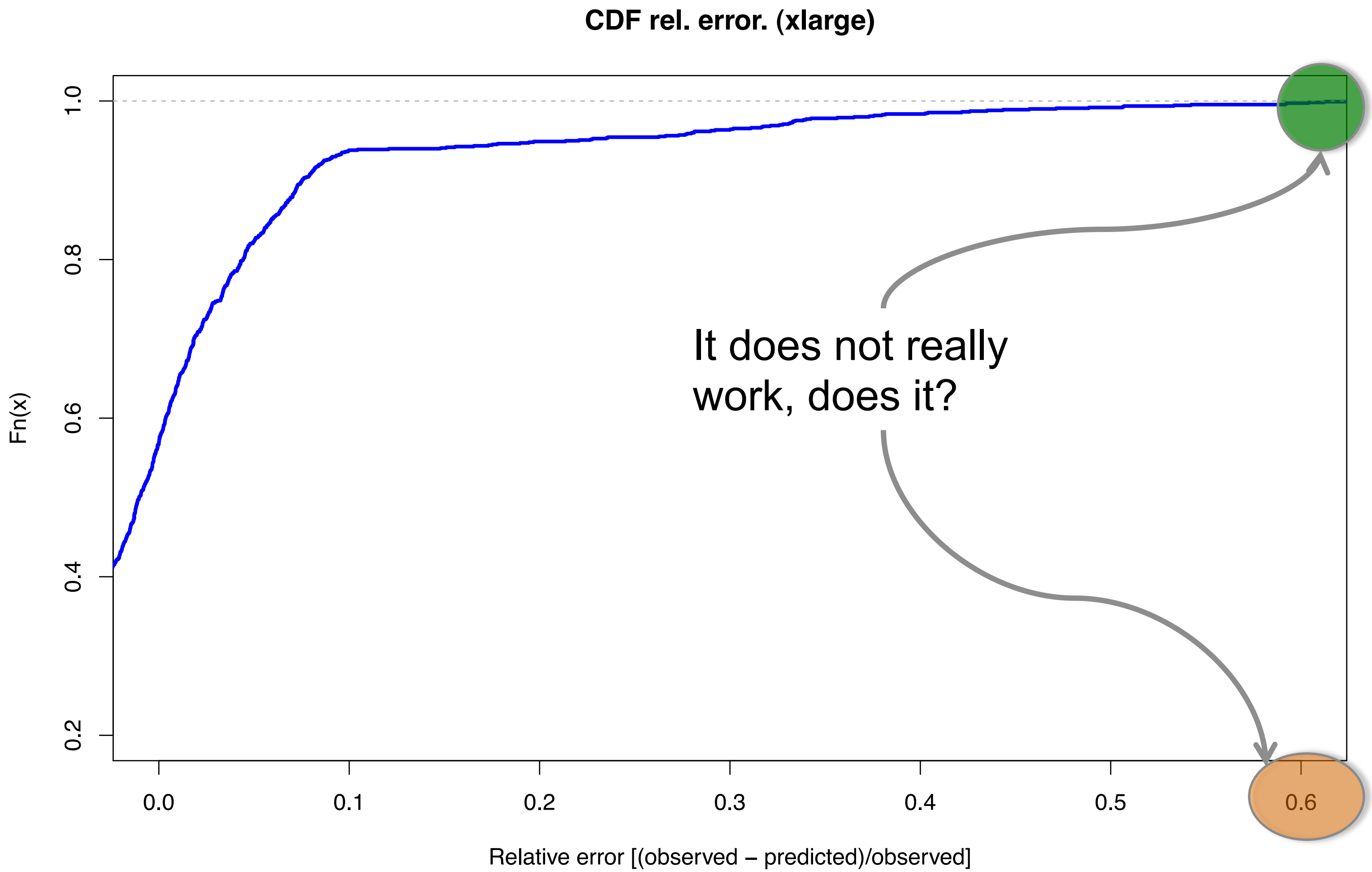
m1.large



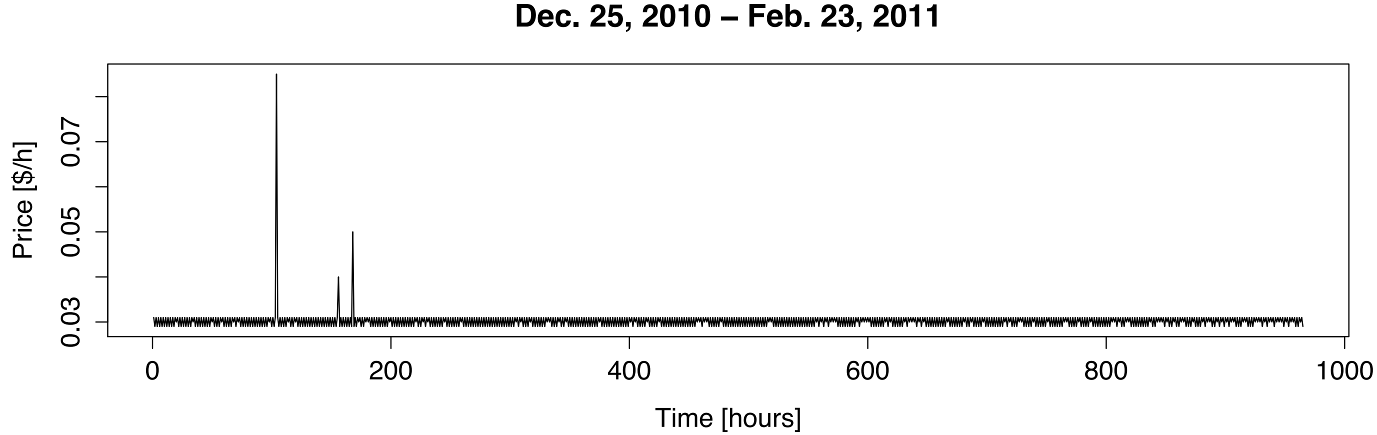
2ND ATTEMPT (2)

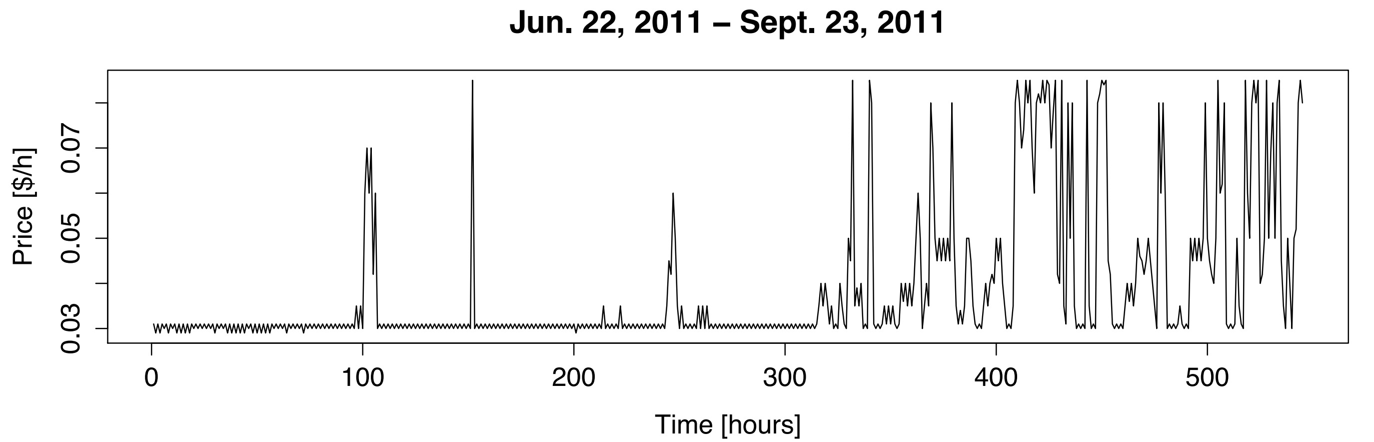


2ND ATTEMPT (3)



ONE MORE THING...





- ... they look quite different, don't they?

3^{RD} ATTEMPT

1. **Employ the autocorrelation function (ACF) to determine the similarity between prices as a function of the time difference between them (*lag*)**

Value at time *t*

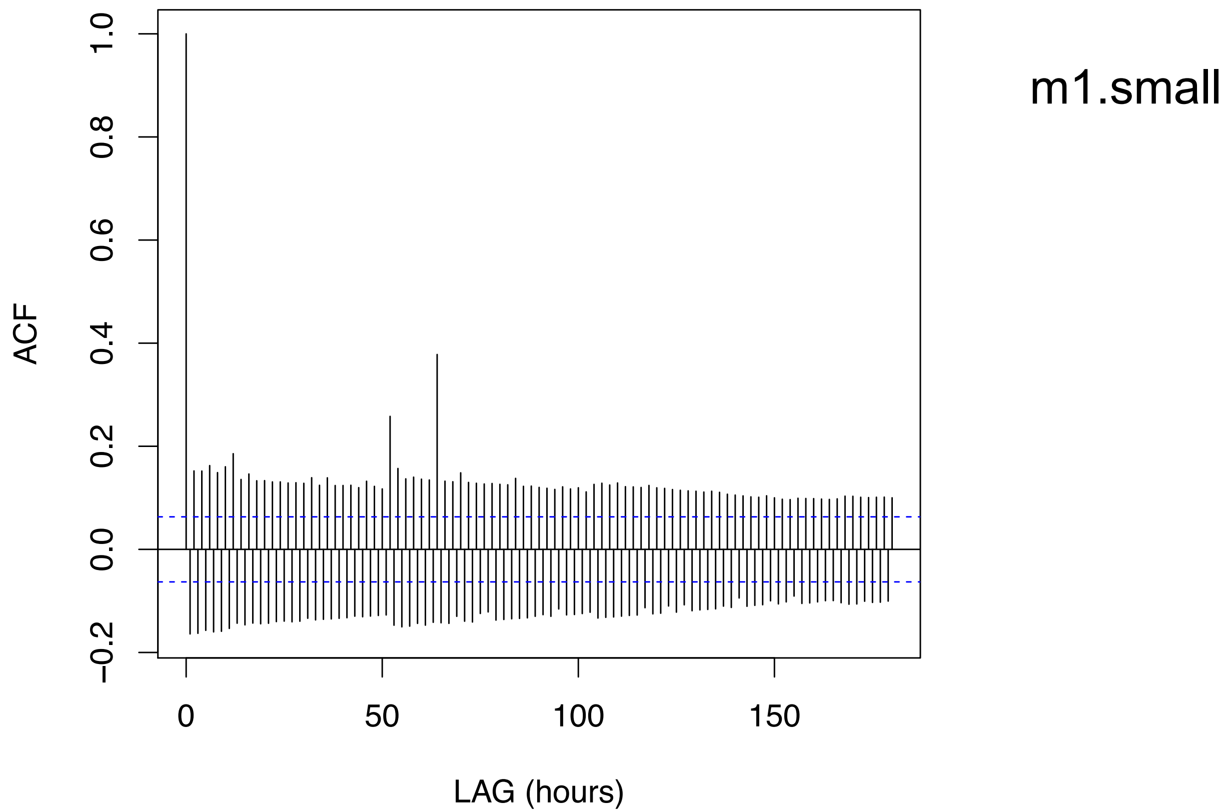
Lag

Average value

No. of observations

$$ACF(l) = \frac{\sum_{t=1}^{N}(x_t - \overline{x})(x_{t+l} - \overline{x})}{\sum_{t=1}^{N}(x_t - \overline{x})^2}$$

2ND ATTEMPT

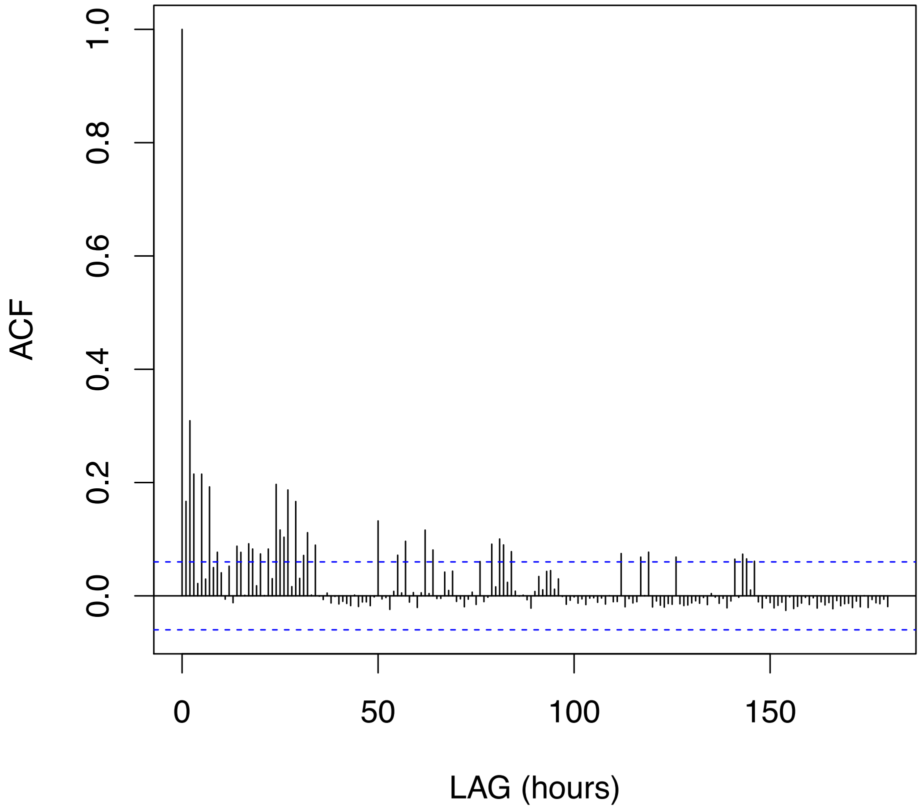


- **The autocorrelation function (ACF) of the prices confirms the lack of any relationship between prices over the time**

2. Use a normal approximation to model prices

- There is some evidence that prices in similar markets are approximately normally distributed

2ND ATTEMPT



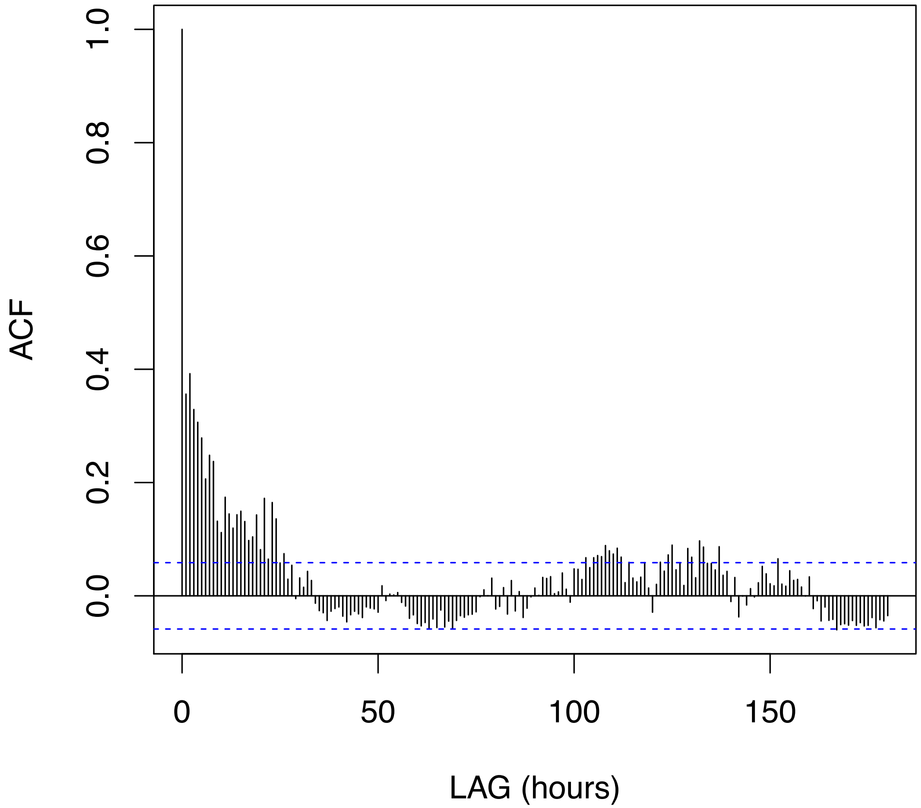
m1.large

- **The autocorrelation function (ACF) of the prices confirms the lack of any relationship between prices over the time**

2. Use a normal approximation to model prices

- There is some evidence that prices in similar markets are approximately normally distributed

2ND ATTEMPT



m1.xlarge

- **The autocorrelation function (ACF) of the prices confirms the lack of any relationship between prices over the time**

2. Use a normal approximation to model prices

  - There is some evidence that prices in similar markets are approximately normally distributed

SPOT PRICE DISTRIBUTION

	Data	Normal Approx.
Minimum	0.029	0.02098
1^{st} Quartile	0.029	0.02877
Median	0.031	0.03020
3^{rd} Quartile	0.031	0.03163
Maximum	0.085	0.04010
Mean (1^{st} moment)	0.0302	0.03025
Variance (2^{nd} moment)	4.503941×10^{-6}	4.515323×10^{-6}
Skewness (3^{rd} moment)	1.877202×10	1.659259×10^{-2}
Kurtosis (4^{th} moment)	4.700735×10^{2}	3.004374

m1.small (Linux/Unix), us-east1 region. Dec. 25, 2010 – Feb. 23, 2011

The distribution of the prices is more heavy tailed (confirmed also by Q-Q plots and Shapiro-Wilk test)

PRICE PREDICTION ALGORITHM

① **Collect price statistics, and compute mean and variance**

② **Compute *ACF* of the prices for lag *l*, with *l* being the prediction horizon**

③ **If *ACF(l) > 0.4*,**

a) Predict the future prices using linear regression, $y = a + bx$, otherwise

b) Use the quantile function (inverse CDF) of the Normal distribution to predict the future prices

- The inverse CDF returns the value of x such as $P(X \leq x) = p$

④ **Use the max value returned by (3) as a "limit price"**

⑤ **In case of out-of-bid events, increase the bid by 40%**

PREDICTION ALGORITHM PERFORMANCE

Prediction (hours)	Target availability	Achieved availability	Avg. big ($/h)
6	90%	99.673%	0.03170
6	99%	99.673%	0.03270
6	99.999%	99.673%	0.03463
12	90%	99.675%	0.03207
12	99%	99.675%	0.03307
12	99.999%	99.675%	0.03499
24	90%	99.679%	0.03253
24	99%	99.679%	0.03350
24	99.999%	99.679%	0.03533

m1.small (Linux/Unix), us-east1 region. Dec. 25, 2010 – Feb. 23, 2011.
The price of "on-demand" instances is 0.085$/h

PART 2: MODEL AND SLA (1)

The SaaS provider

1. *Pays the IaaS for renting computing resources*
2. *Charges its customer for executing WS transactions*
3. *Pays penalties to its customer for failing to meet performance and availability objectives*

MODEL AND SLA (2)…

1. For each accepted and completed request, the client pays a charge of c $
2. <u>Performance</u>: if the avg. resp. time, β, over an interval of length *t* exceeds the threshold q, then the provider pays a penalty of r_1 $ for each job executed in the interval
3. <u>Availability</u>: the provider should pay a penalty of r_2 $ for each rejected job
4. <u>Disaster</u>: the provider is liable to pay a r_3 $ for every accepted job that is lost due to resources becoming unavailable
5. The provider pays r_4 $/h to rent each server

- The provider tries to optimize the average net revenue earned per unit time

... OR

Prob. that the avg. resp. time exceeds the threshold

Prob. that spot instances are terminated

$$R = \gamma[c - r_1 P(\beta > q)] - r_2(\lambda - \gamma) - r_3 P(l_p < s_p) L - r_4 n$$

Rate at which jobs are rejected

Avg. no. of jobs in the system

- **r_1 = penalty for "performance"**
- **r_2 = penalty for "availability"**
- **r_3 = penalty for "disaster"**
- **r_4 = server rental cost**
- **γ = rate at which jobs are accepted into the system**

POLICIES

Resource allocation and admission control

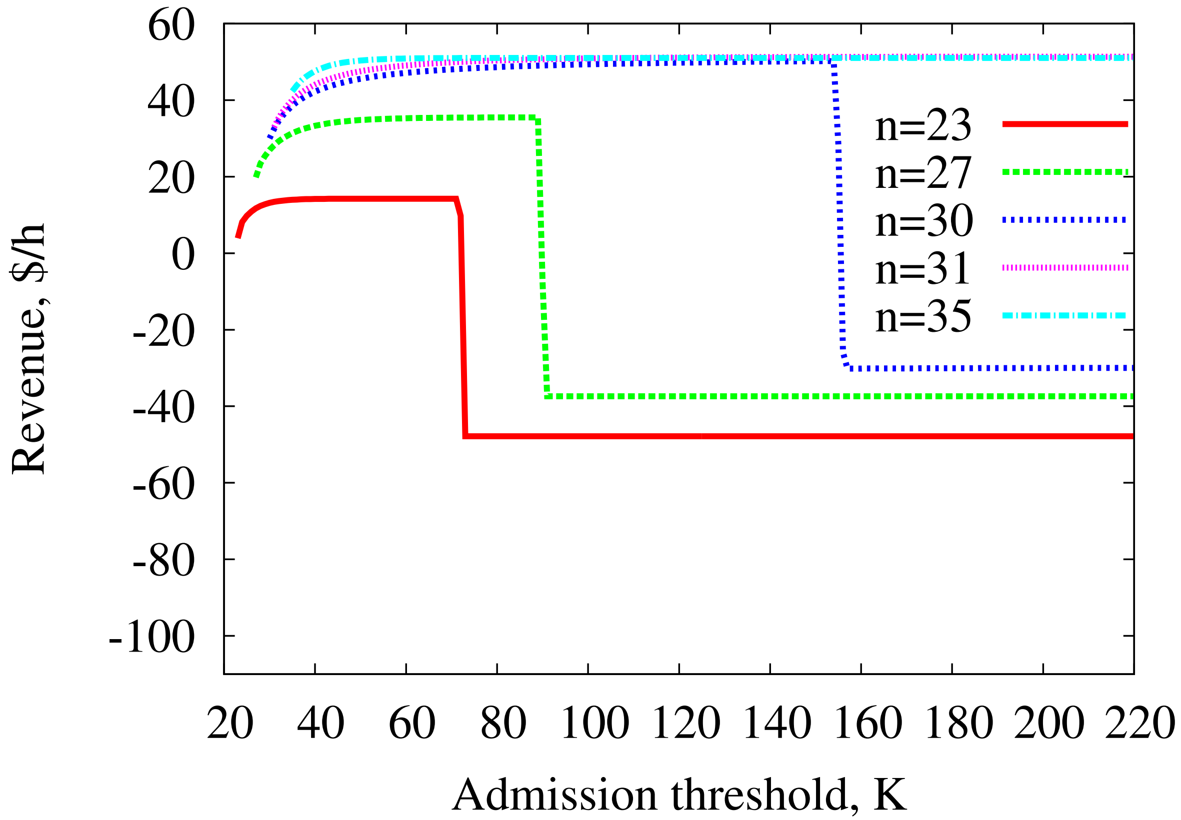
- Admission control defined by means of a threshold, K

1. **"Threshold" policy**
   - The simultaneous optimization of K and n is non trivial
   - Treats the system as an $M/M/n/K$ queue
   - Uses a "Hill Climbing" algorithm to find the "best" value of n and K
2. **"Heuristic": assumes that the response time is dominated by the service time (true in "large" systems)**
   - Treats the system as an $GI/G/n$ queue
   - $K = \infty$, e.g., no job is rejected

PERFORMANCE EVALUATION - SETTINGS

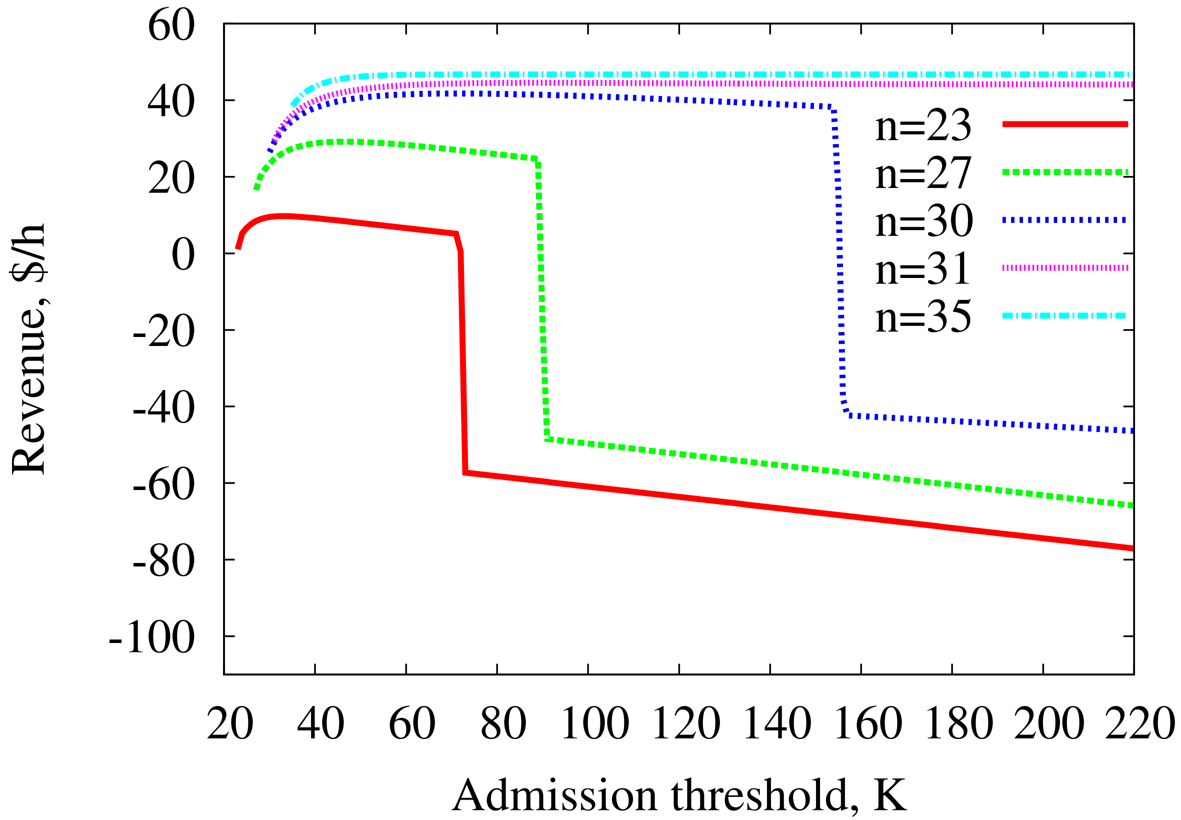
b	0.5 sec.	Average service time
q	1 sec.	Performance threshold
c	2.951×10^{-5} $	Charge per job
r_1	$1.5 \times c$	Penalty (performance)
r_2	$2 \times c$	Penalty (availability)
r_3	$3 \times c$	Penalty (disaster)
r_4	0.085 $/h	Rental cost (“on-demand” instances)
t	1 min.	Interval length (SLA evaluation)

REVENUE AS FUNCTION OF K AND N



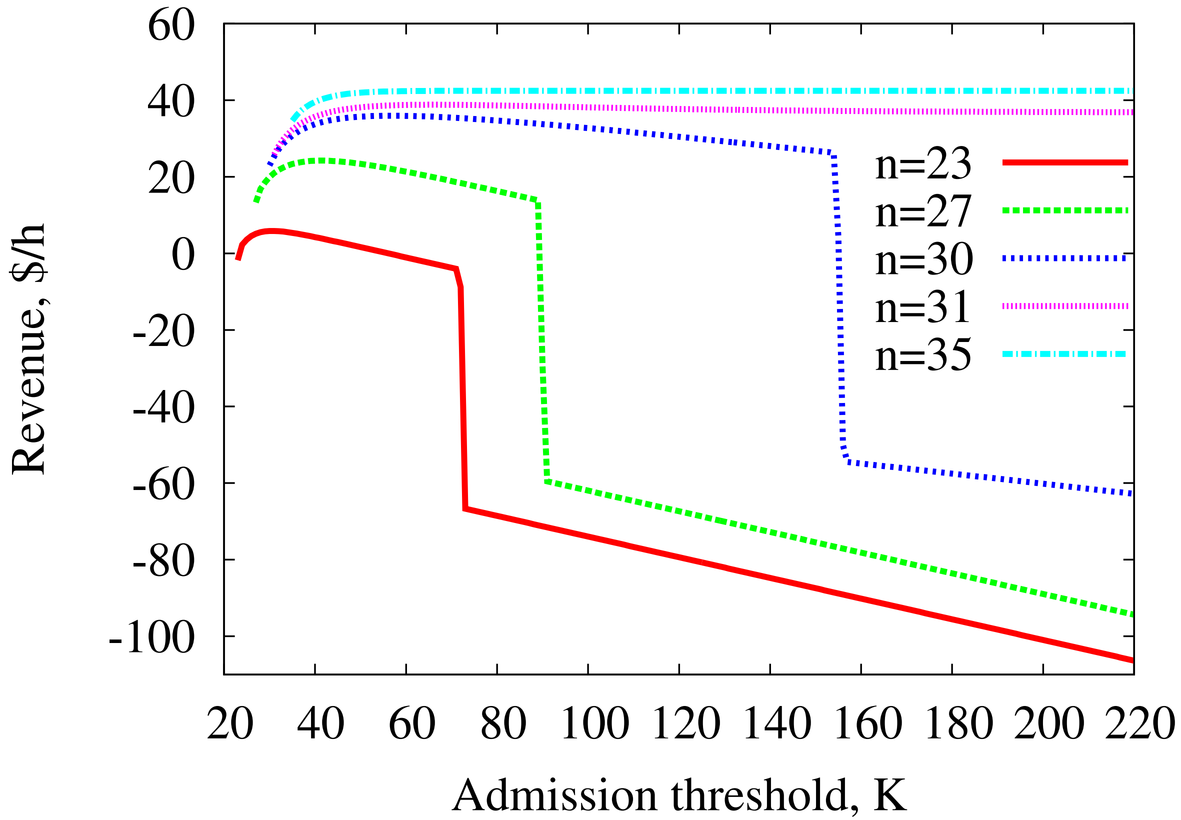
$p(l_p < c_p) = 0$

REVENUE AS FUNCTION OF K AND N



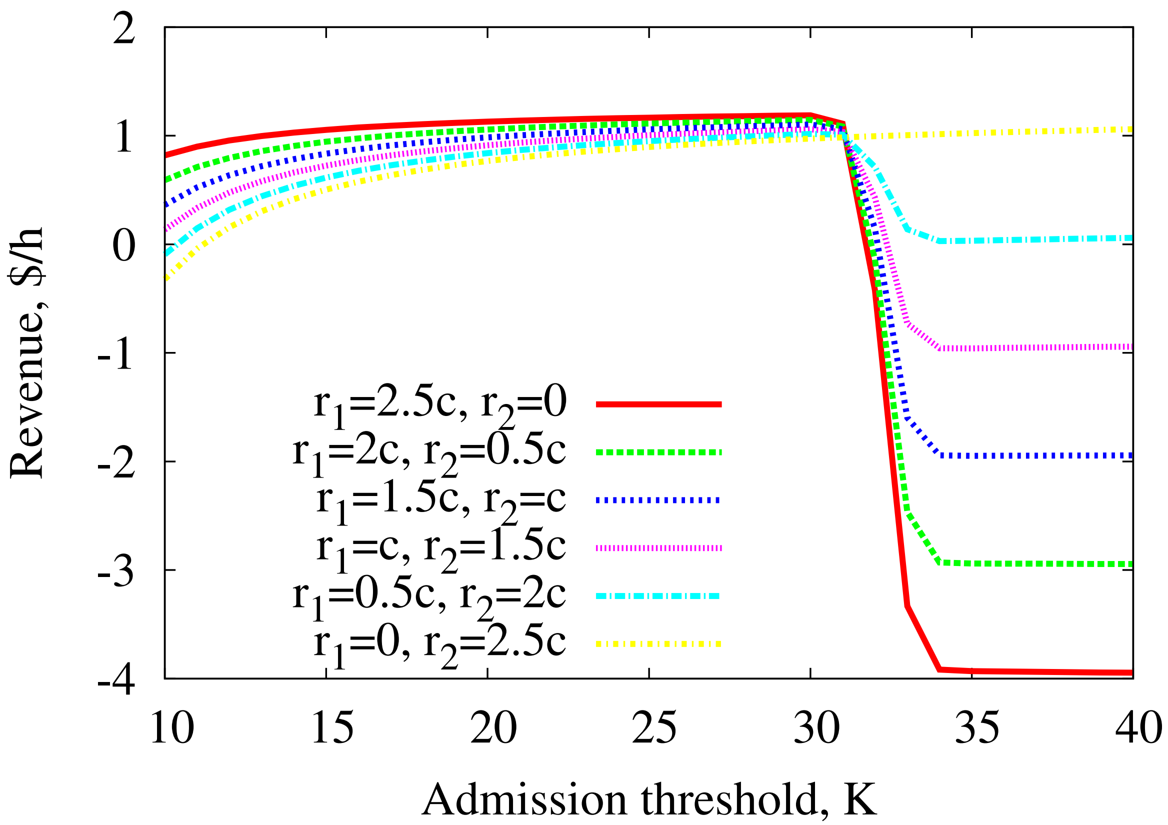
$p(I_p < c_p) = 0.25$

REVENUE AS FUNCTION OF K AND N



$p(l_p < c_p) = 0.5$

REVENUE AS FUNCTION OF R_1, R_2 AND K

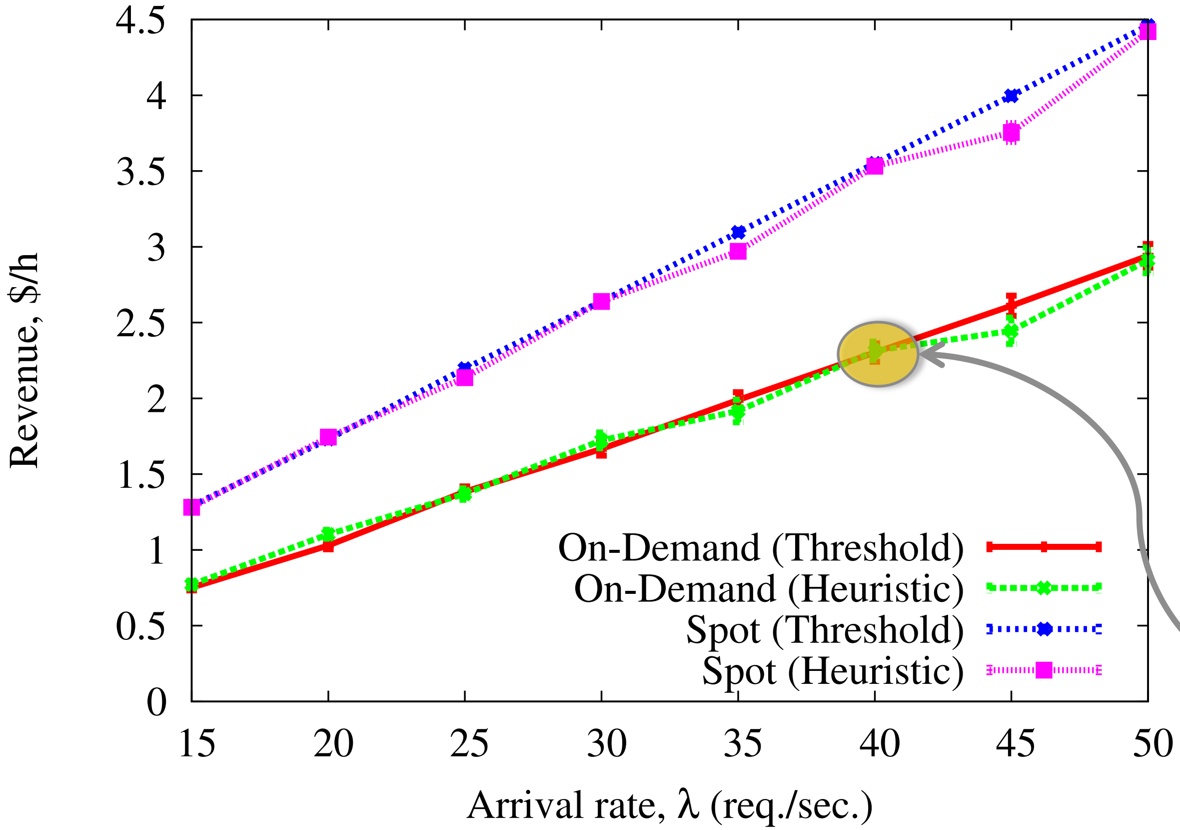


The system is saturated

- Have to choose between a long queue and rejecting many jobs

$n = 10$, $\rho = 10$, $p(l_p < c_p) = 0$ (e.g., no disasters occur)

REVENUE AS FUNCTION OF THE ARRIVAL RATE



1. The revenue increases with the load
2. The use spot instances makes the system more profitable
3. The “Heuristic” policy performs almost as good as the “Threshold” algorithm

Each point represents a run lasting 28 days

$ca^2=1$, $cs^2=1$, $\rho = 7.5,\ldots,25$, $p(l_p < c_p) = 0.001$

CONCLUSIONS

- **I have discussed an approach aiming at maximizing the net revenue earned by a SaaS using spot instances to provide a web service to paying customers**
  1. How much to bid for resources on the spot market?
  2. How many servers should be allocated for a given time period, and how many jobs to accept?

- **The number of running servers, as well as the maximum number of jobs admitted into the system, have a significant effect on the earned revenues**
  - The optimal queue length is highly dependent on the availability level (e.g., shorter queue when the likelihood premature termination is high)