



## CS 475: Performance Evaluation

Wim Bohm & Sanjay Rajopadhye  
Colorado State University  
Fall 2012



## Analyzing Program Performance

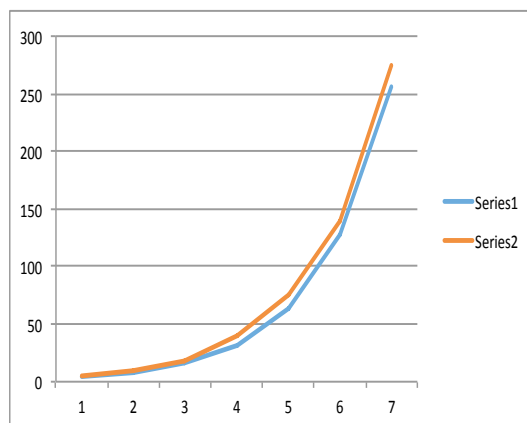
- In empirical Computer Science, we plot functions describing the run time (or the memory use) of a program:
  - This can be as a **function of the input size**. We have seen this in e.g. cs320 or cs420, where we study polynomial and exponential (**monotonically growing**) sequential complexity.
  - In this class we also study program performance as a function of the number of processors.
    - In this case the functions are positive and, hopefully decreasing.
    - Also we plot speedup curves, which are usually asymptotic

## Analyzing/Plotting Data

- When you run a program for a number of inputs ( $n$ ) on a parallel machine with a number of processors ( $p$ ), you end up with performance data sets. You want to characterize these in (a set of) functions:  
x: input size, y: performance or  
x: #processors, y: performance.
- To study (parallel) program's performance, we often use plotting tools
  - gnuplot, excel, matlab ... (in these slides: excel)
- Let's look at increasing functions first.

### Ungraded quiz (survey)

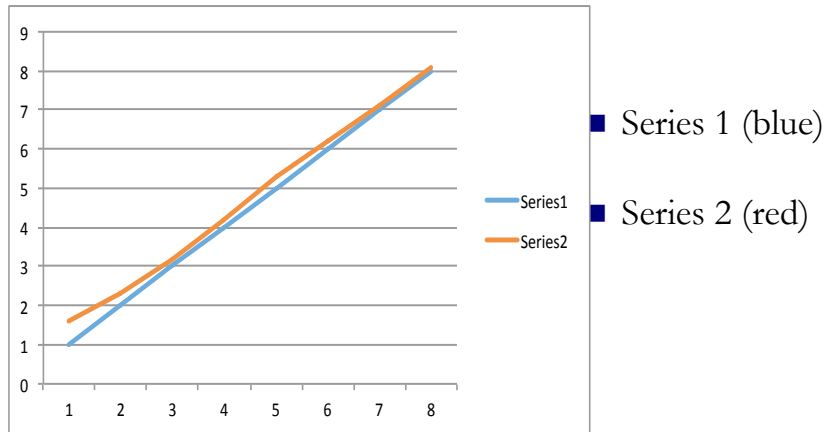
One plot below is ideal, the other is "just a bit off."  
Which one is the ideal?



- Series 1 (blue)
- Series 2 (red)

### Ungraded quiz (survey)

One plot below is ideal, the other is “just a bit off.”  
Which one is the ideal?

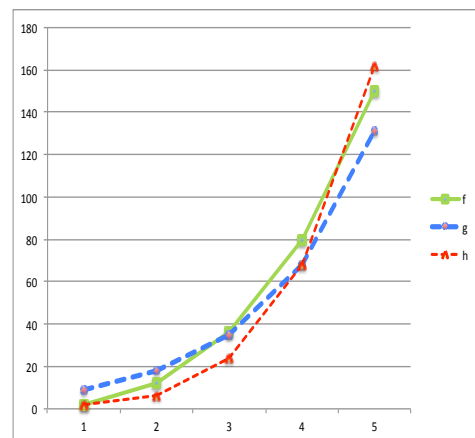


### Example: 3 data sets f, g and h

n	f(n)	g(n)	h(n)
1	2	9	2
2	12	18	6
3	36	35	24
4	80	68	68
5	150	131	162

What kinds of functions are f, g and h?

- exponential? which base?
- polynomial? which order?



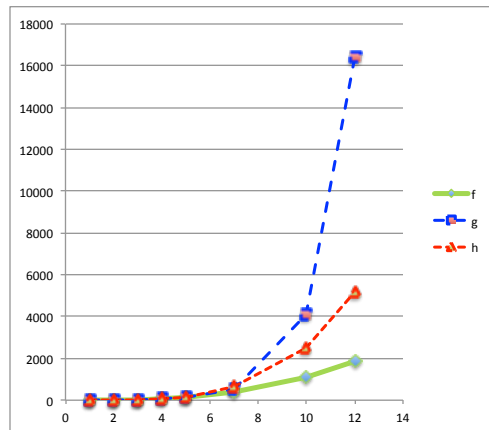
**Hard / impossible to infer**

# Why are functions hard to infer?

- Two problems:
  - Very small domain (here 1..5)
    - Try to get a large data domain
  - Interpreting super-linear functions from plots is hard
    - All polynomials and exponentials **swoop up**

## Larger domain

n	f(n)	g(n)	h(n)
1	2	9	2
2	12	18	6
3	36	35	24
4	80	68	68
5	150	131	162
7	400	520	624
10	1100	4106	2510
12	1872	16396	5196



Do you get a better idea now?

Which function may be polynomial, which exponential?

Still, not all clear (order, base...), h(n) may spike up later...

## Straight Lines

We get the most information from **straight lines!**

- We can easily recognize a straight line ( $y = ax+b$ )
  - The **slope (a)** and **y intercept (b)** tells us all.
- **So we need to turn our data sets into straight lines.**
- This is easiest done using log-s, because they turn a **multiplicative factor into a shift** (y axis crossing b) , **and an exponential into a multiplicative factor** (slope a)

## Exponential functions

- $\log(2^n) = n \log 2$     linear in n
- $\log(3^n) = n \log 3$     **angle** of the line: base of log
  
- $\log(4 \cdot 3^n) = n \log 3 + \log 4$     **/4 shifts up**
- $\log((3^n)/4) = n \log 3 - \log 4$     **/4 shifts down**

## Exponentials: semi-log plot

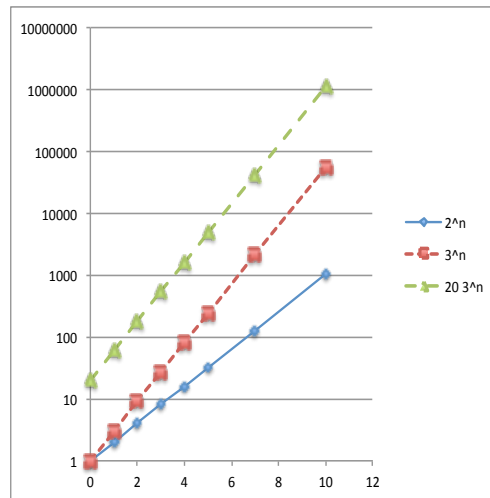
n	2 <sup>n</sup>	3 <sup>n</sup>	20*3 <sup>n</sup>
0	1	1	20
1	2	3	60
2	4	9	180
3	8	27	540
4	16	81	1620
5	32	243	4860
7	128	2087	41740
10	1024	56349	1126980

semi-log plot:

y-axis on log scale  
x-axis linear

angle: base

shift: multiplicative factor



## Polynomials

- What if we take the log of a polynomial (actually a monomial)?

e.g.  $f(n) = 5n^3$

$$\log(f(n)) = \log(5n^3) = \log 5 + 3 \log(n)$$

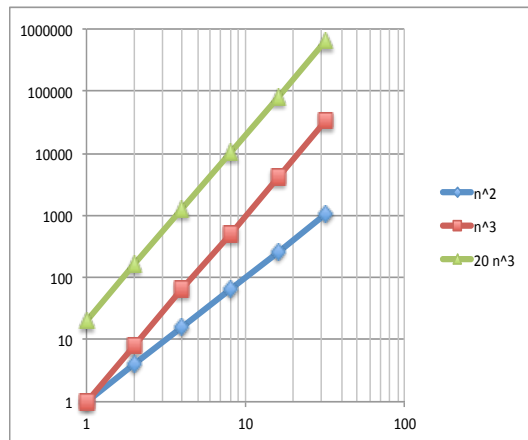
not a straight line!

- But the **log of a monomial is linear in log(n)**
- Therefore we need to plot polynomials on a log-log scale (both x and y axis logarithmic)

## Polynomials: log-log plot

n	n <sup>2</sup>	n <sup>3</sup>	20*n <sup>3</sup>
1	1	1	20
2	4	8	160
4	16	64	1280
8	64	512	10240
16	256	4096	81820
32	1024	32768	655360

angle: degree  
shift: multiplicative factor



## logs of sums

- Often we don't have a single factor in our function:

- $3^n + 2^n$

- $n^3 + n^2$

- **Watch it:** log of sum is not sum of logs (what is?)

- Straight lines not completely straight anymore but **asymptotically straight:**

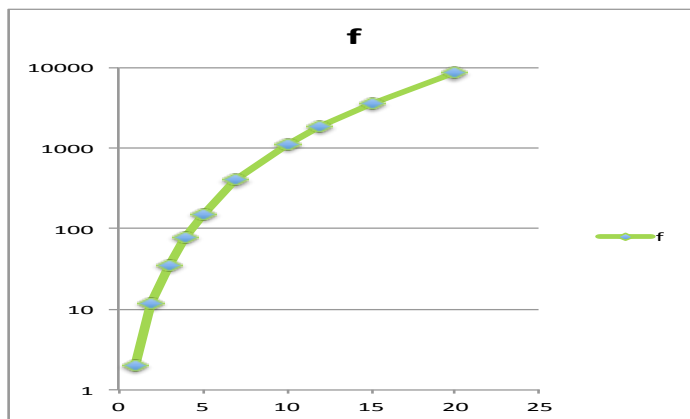
$$\log(3^n + 2^n) = \log((1 + (2/3)^n)3^n) = \log(1 + (2/3)^n) + n\log(3)$$

$$\log(n^3 + n^2) = \log((1 + (1/n))n^3) = \log(1 + (1/n)) + 3\log(n)$$

$\log(1 + (2/3)^n)$  and  $\log(1 + (1/n))$  go to zero for large n

## Back to the data: $f$

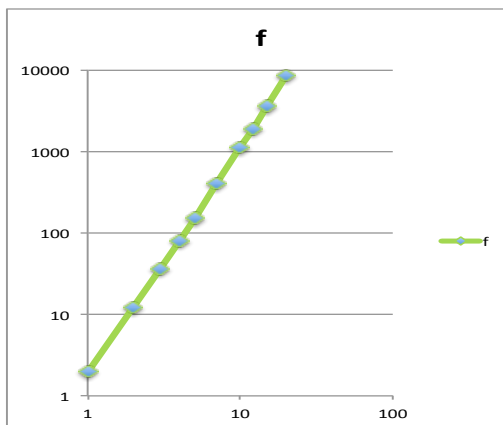
$n$	$f(n)$
1	2
2	12
3	36
4	80
5	150
7	400
10	1100
12	1872



The semi-log plot does not give a straight line, so  $f$  is not exponential

## Is $f$ polynomial?

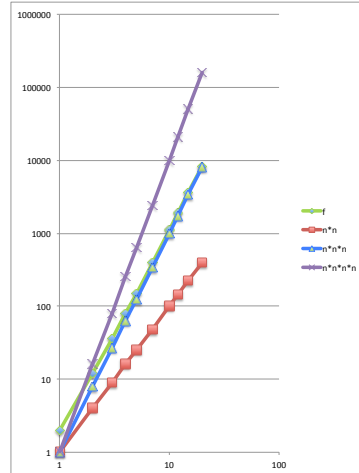
$n$	$f(n)$
1	2
2	12
3	36
4	80
5	150
7	400
10	1100
12	1872



YES! The log-log plot goes asymptotically to a straight line, so  $f$  is polynomial, but what is its leading term?

## What is f's degree?

n	f(n)	n <sup>2</sup>	n <sup>3</sup>	n <sup>4</sup>
1	2	1	1	1
2	12	4	8	16
3	36	9	27	81
4	80	16	64	256
5	150	25	125	625
7	400	49	343	2401
10	1100	100	1000	10000
12	1872	144	1728	20736

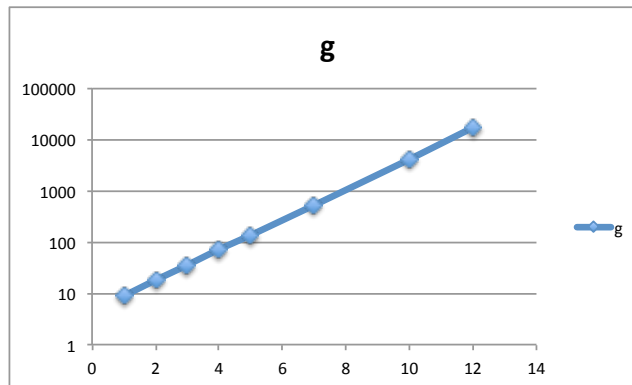


Compare with  $n, n^2, n^3, n^4$

f is degree 3, no multiplicative factor (no shift up):  $f(n)=n^3+..$   
 We usually only worry about the leading term.

## How about g?

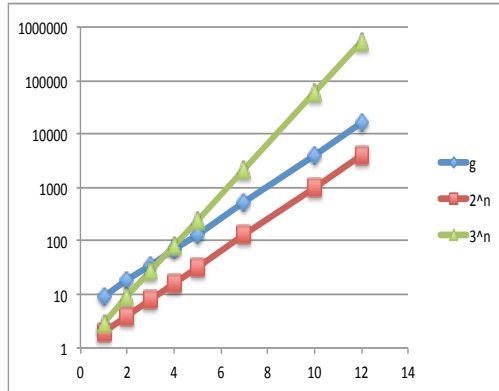
n	g(n)
1	9
2	18
3	35
4	68
5	131
7	520
10	4106
12	16396



g is linear on semi log plot so exponential  
 what base: compare to  $2^n, 3^n$

## How about g?

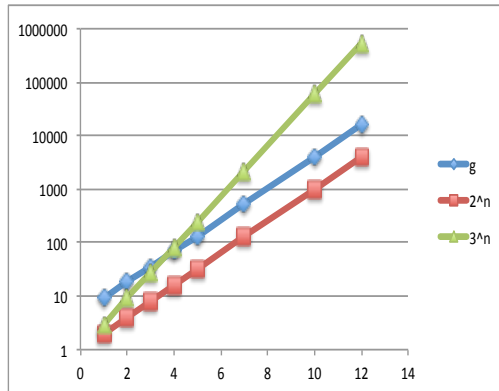
n	g(n)	2 <sup>n</sup>	3 <sup>n</sup>
1	9	2	3
2	18	4	9
3	35	8	27
4	68	16	81
5	131	32	243
7	520	128	2187
10	4106	1024	59049
12	16396	4096	531441



g is linear on semi-log plot so exponential  
 what base: compare to 2<sup>n</sup>, 3<sup>n</sup>

## How about g?

n	g(n)	2 <sup>n</sup>	3 <sup>n</sup>
1	9	2	3
2	18	4	9
3	35	8	27
4	68	16	81
5	131	32	243
7	520	128	2187
10	4106	1024	59049
12	16396	4096	531441



g: same slope as 2<sup>n</sup>, but shifted up, factor 4  
 so  $g(n) = 4 \cdot 2^n + \dots$

## Decreasing functions

- This second class of functions can be used to represent running times of programs as a function of the number of processors.
- Ideally, these functions decrease **hyperbolically**
  - $f(p) = c/p$  time to execute the program with  $p$  processors
  - $f(1) = c$  sequential time
- But this is hardly ever the case. One of the reasons for this is that programs have inherently sequential parts, that do not speed up with more processors:
  - $f(p) = a + c/p$   $a$ : the sequential part,  $c$ : the parallelizable part

## Plotting hyperbolic functions

- A simple way to turn  $T(p) = c/p$  into a straight line is to plot its reciprocal:  $y = 1/T(p) = p/c$ 
  - This is a straight line with slope  $1/c$ .
  - When analyzing parallel performance we scale this to  $y' = T(1)/T(p)$ . If  $T(p)$  is the time it takes to execute a program with  $p$  processors, we call this the **speedup of the program**
- In the case of  $T(p) = a + c/p$ , the speedup is
$$S(p) = T(1)/T(p) = (a+c)/(a+c/p)$$
  - For  $a > 0$  this is not a straight, but a curve that grows and then flattens out to a constant  $(a+c)/a$

## Plotting Data: Summary

- Visually, a straight line conveys the most information.
  - If your data is not linear, massage it so that is linear, then deduce the original function.
- If  $y=f(x)$  is polynomial:  $\log y$  is linear with  $\log x$ 
$$y = f(x) = a_0 + a_1x + \dots + a_nx^n \approx a_nx^n \quad (\text{asymptotically})$$
$$\log y = \log a_n + n \log x$$
- If  $y=f(x)$  is exponential:  $\log y$  is linear with  $x$ 
$$y = f(x) = ba^x$$
$$\log y = \log b + x(\log a)$$
- In the case of  $T(p) = a + c/p$ , the speedup is
$$S(p) = T(1)/T(p) = (a+c)/(a+c/p)$$