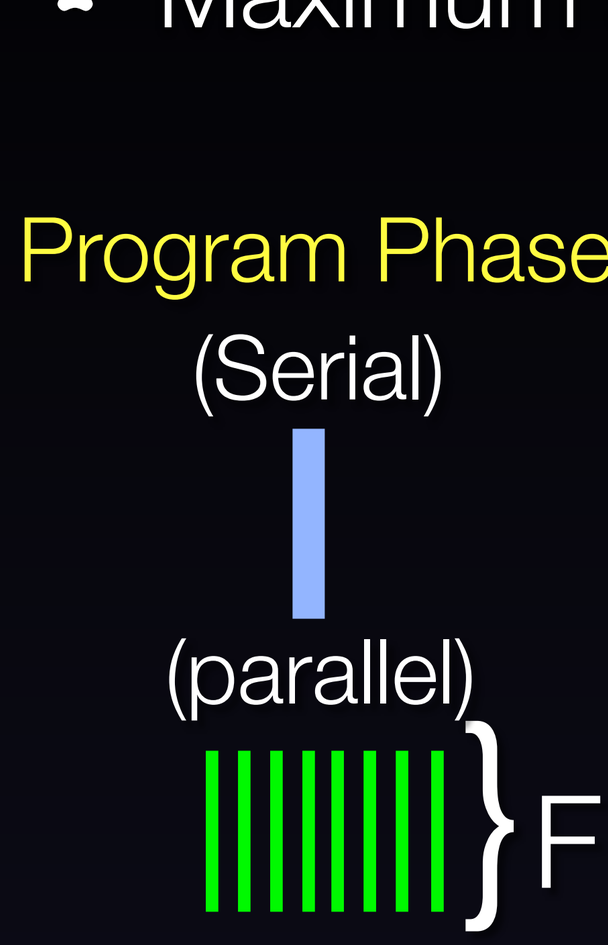
Parallelism and Heterogenity Scaling Laws

Amdahl's Law [1967, Gene Amdahl]

- Maximum speedup achievable on a multicore



Time on 1 core = $\frac{1-F}{1} + \frac{F}{1}$

Time on N cores = $\frac{1-F}{1} + \frac{F}{N}$

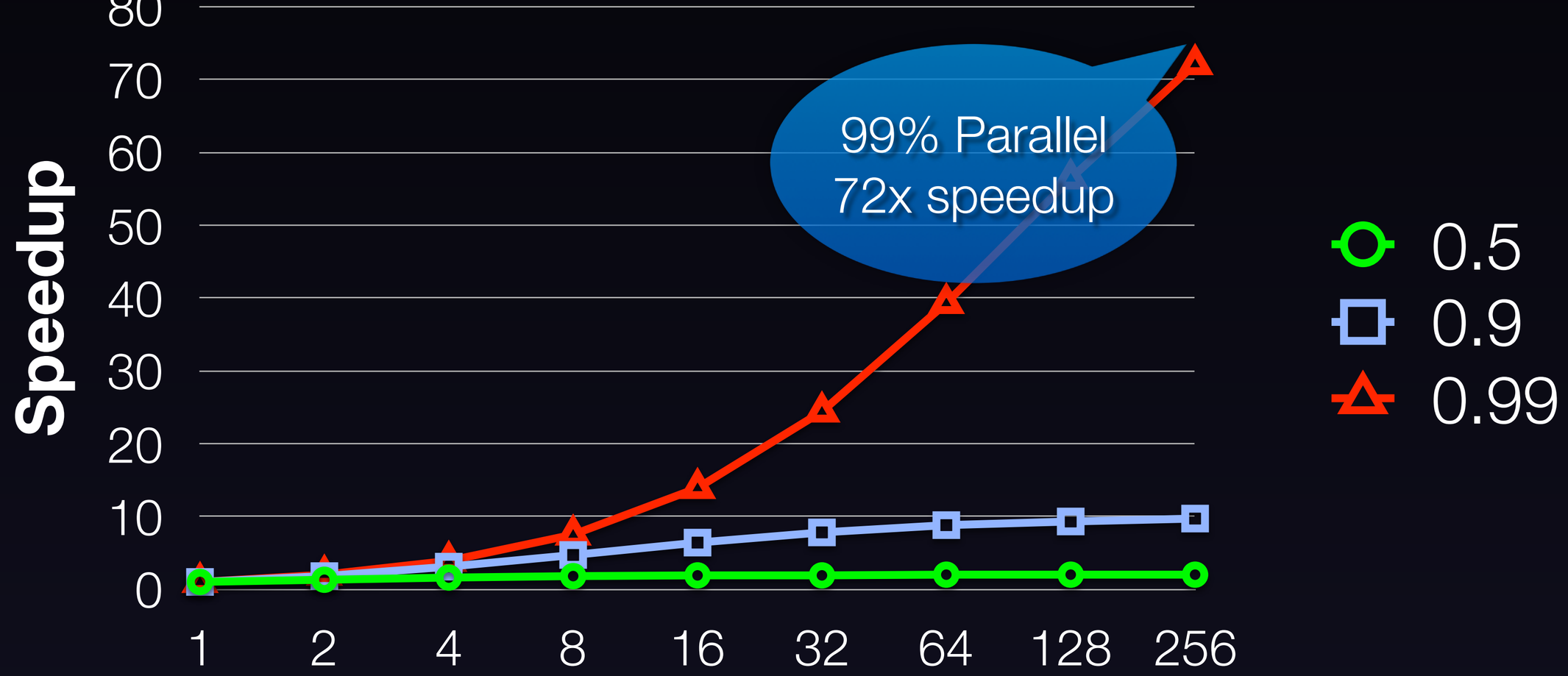
Serial
No

Speedup = $\frac{1}{\frac{1-F}{1} + \frac{F}{N}}$

If F = 0.35
@ 4 cores, speedup = 2
@ ∞ cores, speedup = 3

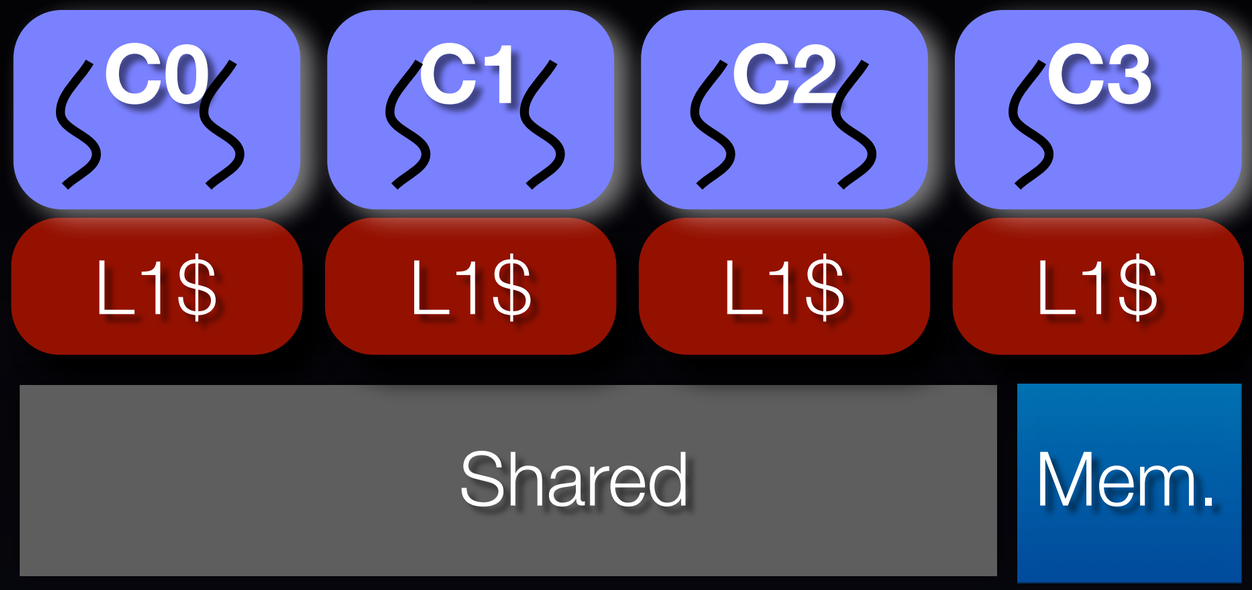
Strong scaling vs Weak scaling

- Strong Scaling : If new machine has K times more resources, how much does perf. improve ?
- Weak Scaling : If new machine has K times more resources, can we solve a bigger problem size ?



Amdahl's Law for Multicores

[Marty and Hill, 2009]



- Multicore Chip partitioned into
  - multiple cores (includes L1 cache)
  - uncore (Intel terminology for Shared L2 cache, L3)

- Resources per-chip bounded
  - Area, Power, $, or a combination
  - Bound of total N resources per-chip.
  - How many cores ? How big each ?

Core Types

- Your favorite trick can be used to improve single-core performance using same resource
  - becoming increasingly hard to do power-efficiently

- Wimpy Core :
  - Consumes 1 CU (CU: measure of core resources)
  - performance = 1

- Hulk Core:
  - consumes R CUs
  - performance = perf(R)

Hulk Cores

- If Perf (R) >= R ; always use the hulk cores.
  - speeds up everything

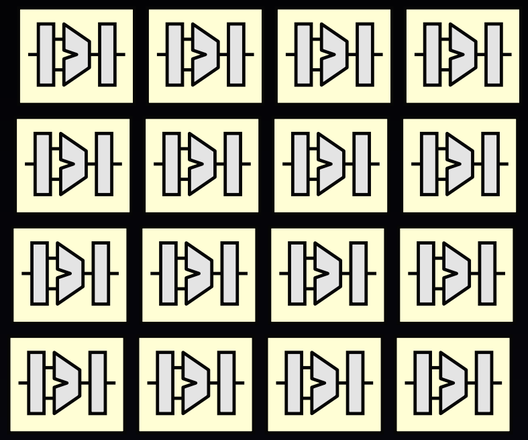
- Unfortunately, life isn't easy Perf (R) < R

- Assume Perf (R) = $\sqrt{R}$
  - reasonable assumption?
  - Microprocessor examples seem to indicate

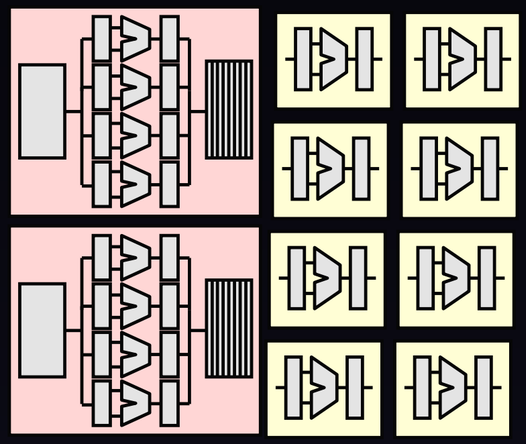
- How to design core for specific Perf (R)
  - basic idea: do many instructions in parallel

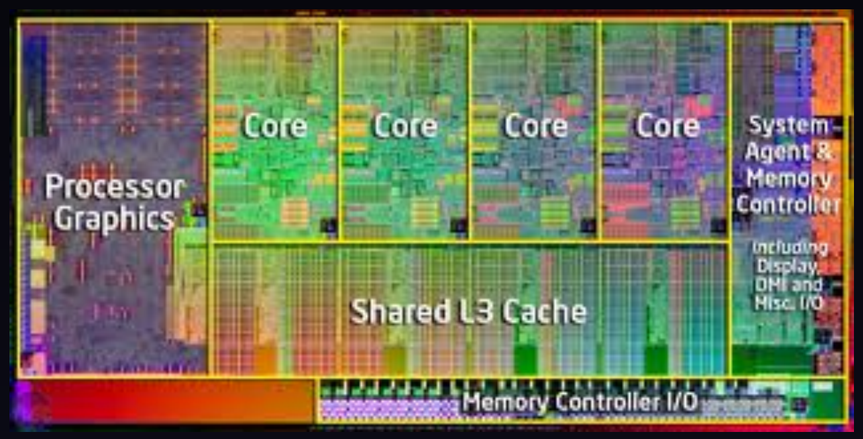
Multicores under consideration

Symmetric

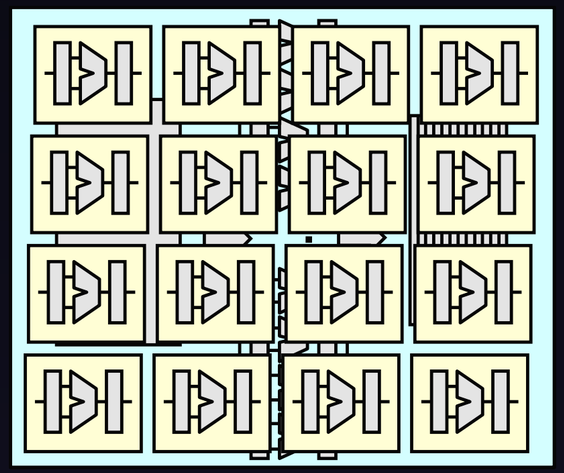


Asymmetric

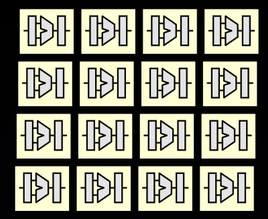




Morphing

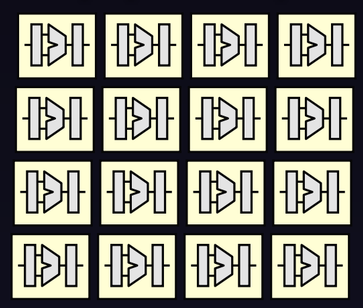


Symmetric Multicores

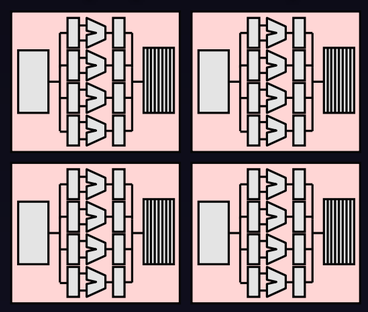


- How many cores ? How big each core ?
- Chip is bounded to N CUs
  - each core has R CUs
- Number of cores per-chip = N/R
- For example, lets say N = 16

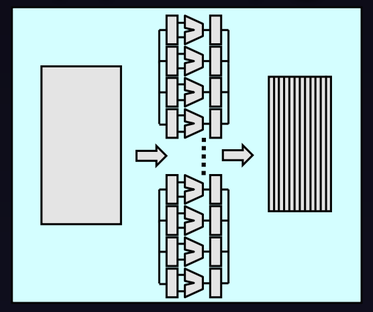
R = 1



R = 4



R = 16



Symmetric Multicore : Performance

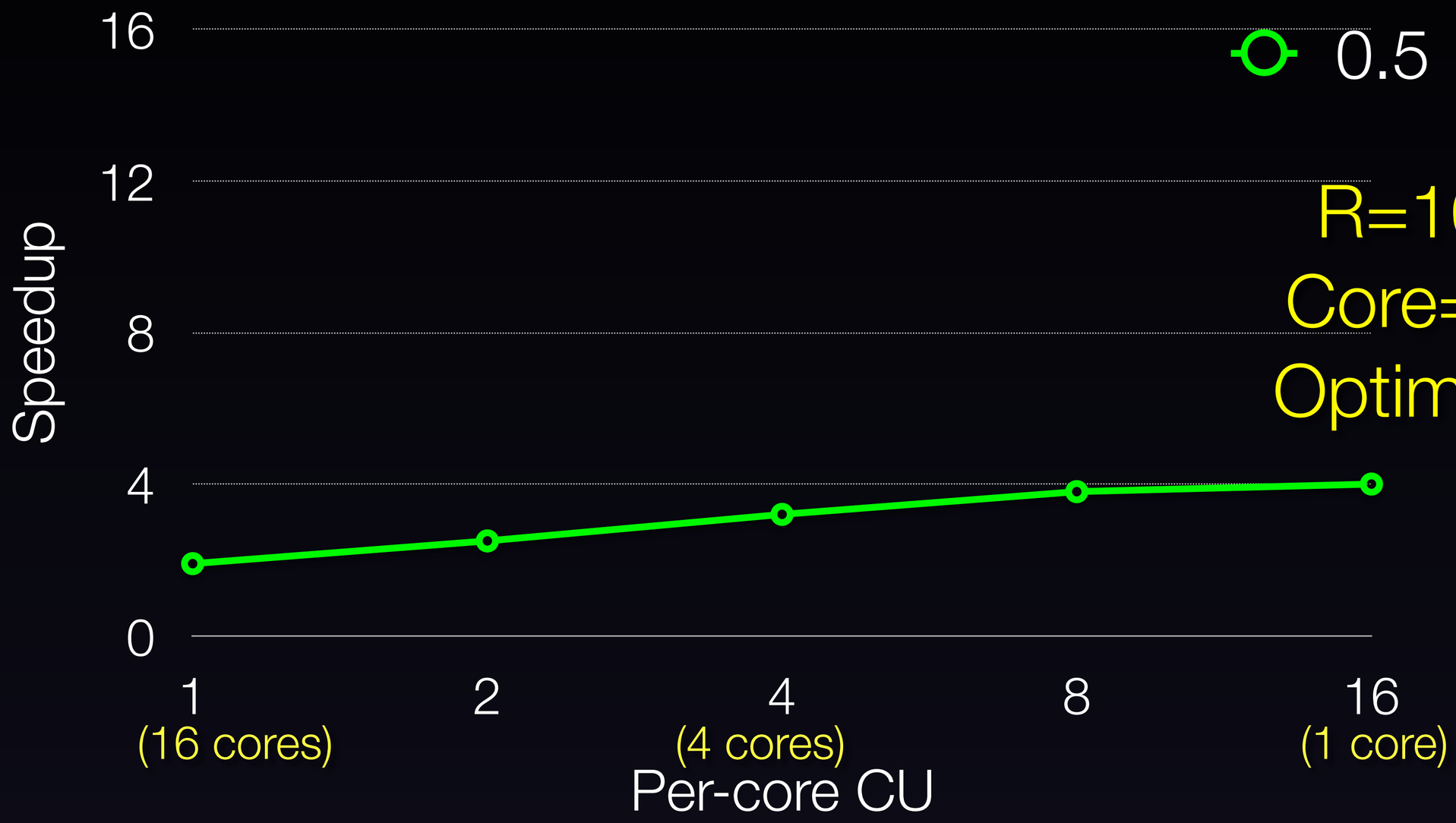
- Serial Phase (1-F) runs on 1 thread on 1 core
  - performance $\propto$ Perf (R)
  - Execution time = (1-F) / Perf (R)
- Parallel Phase uses all N/R cores. Core @ Perf (R)
  - Execution time = F / [Perf (R) * N/R]

$$\text{Speedup} = \frac{1}{\frac{1-F}{Perf(R)} + \frac{F*R}{Perf(R)*N}}$$

Serial Phase Perf(R) ↑

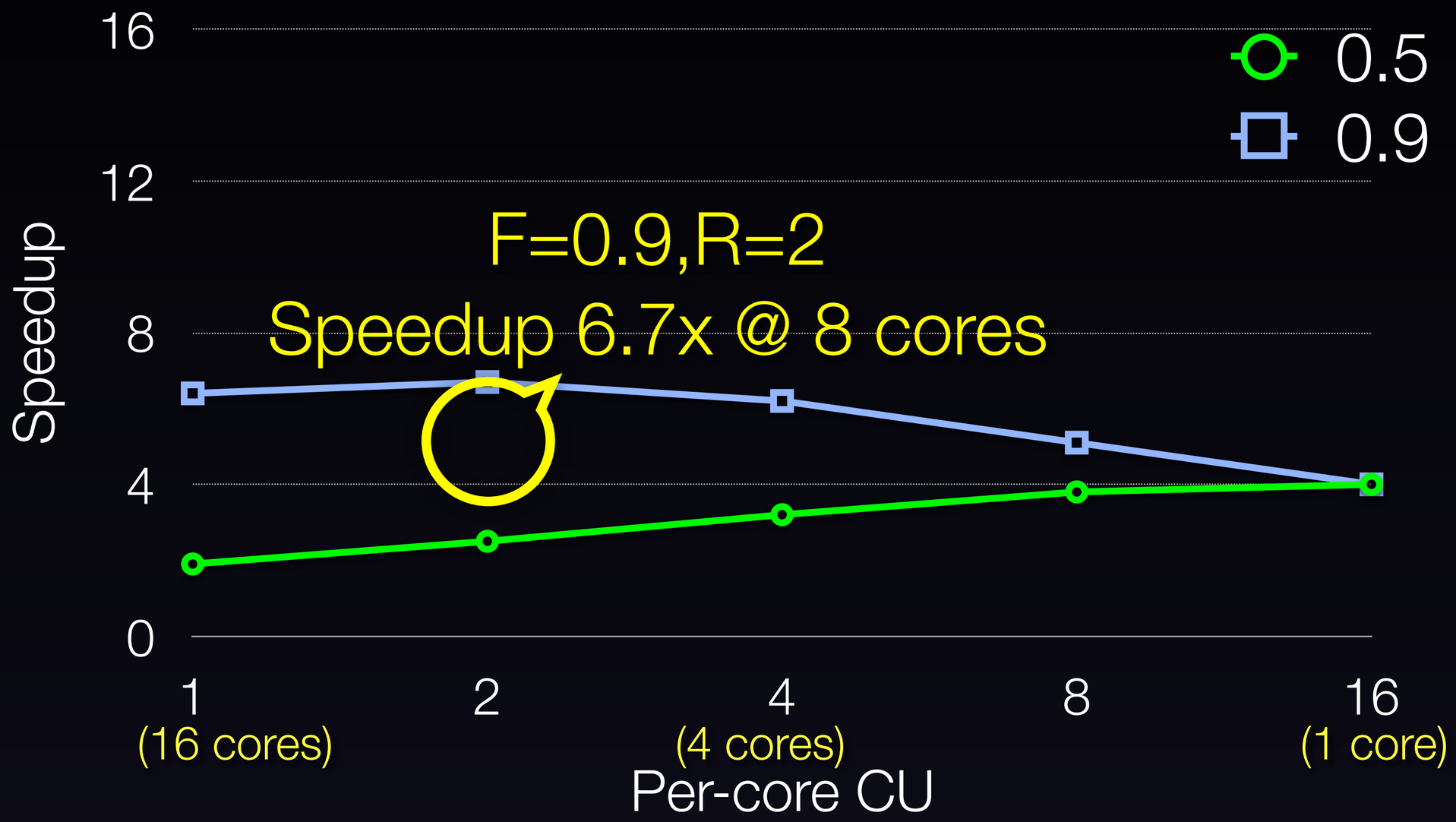
II Phase More cores!

Symmetric Multicore (Chip = 16 CUs)



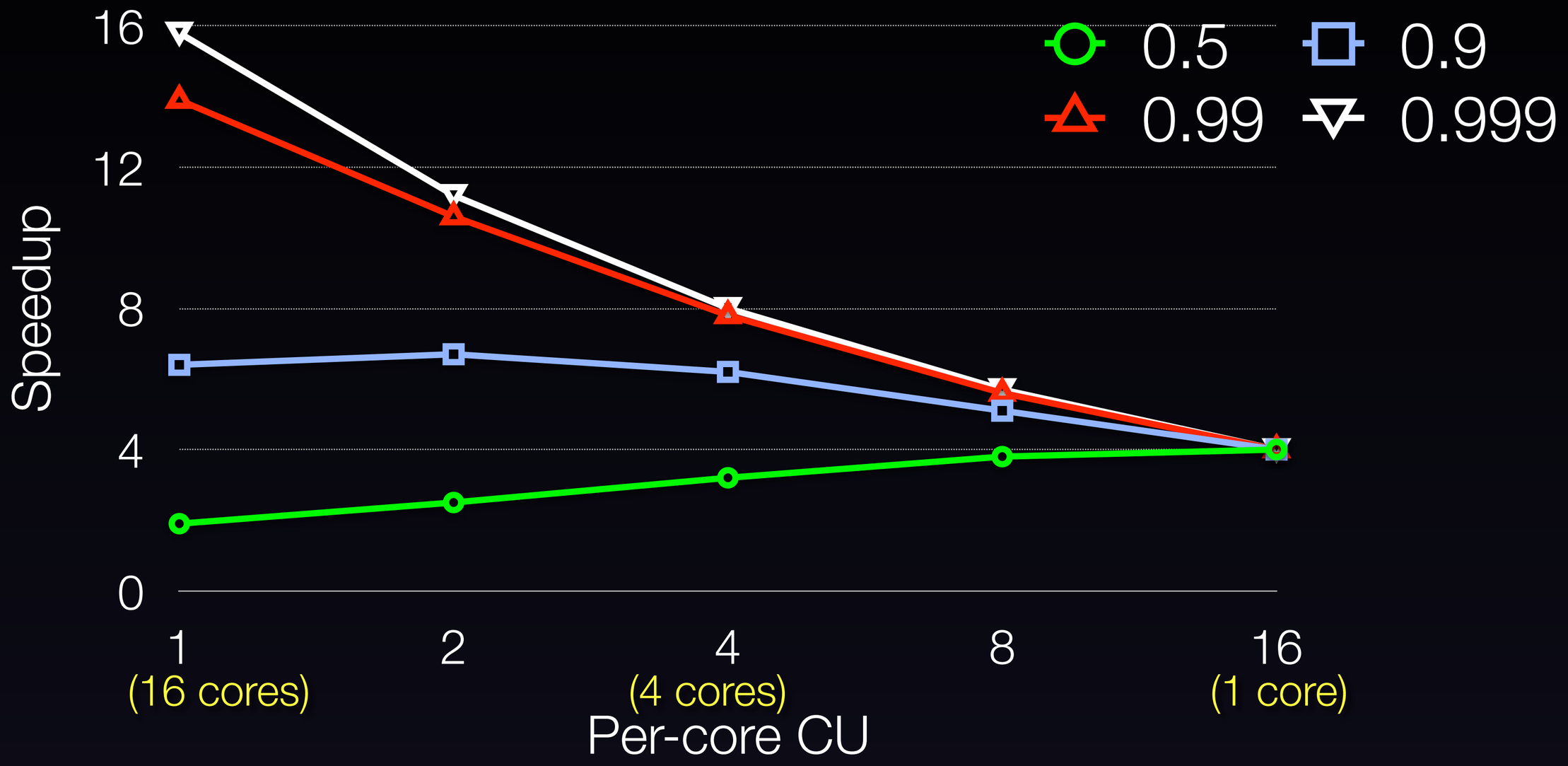
- Need lots of parallelism in multicore world!

Symmetric Multicore (Chip = 16 CUs)



- More parallelism helps; but limited speedup!

Symmetric Multicore (Chip = 16 CUs)



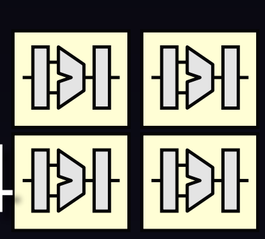
- Applications with high F;
  - significant performance loss with bigger cores
  - Performance loss $\propto \frac{R}{\sqrt{R}} = \sqrt{R}$

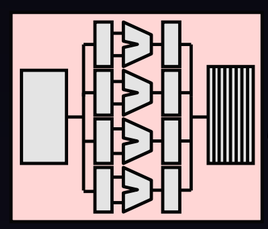
Model-bias towards parallelism

- Remember Perf (R) when scaling up CPU = $\sqrt{R}$
- Lets say 1st gen 1 CU system = 1 CU

- Now consider 2nd gen 4 CU system
  - Four 1CU cores or One 4CU core?
  - When F=0.999; always pick Four 1CU cores

F=0.999
Speedup ~4





Speedup = 2

- Even parallel fraction not perfectly parallel
  - Synchronization, Contention, Locks etc
  - Need SW-Perf(R) (depends on application)

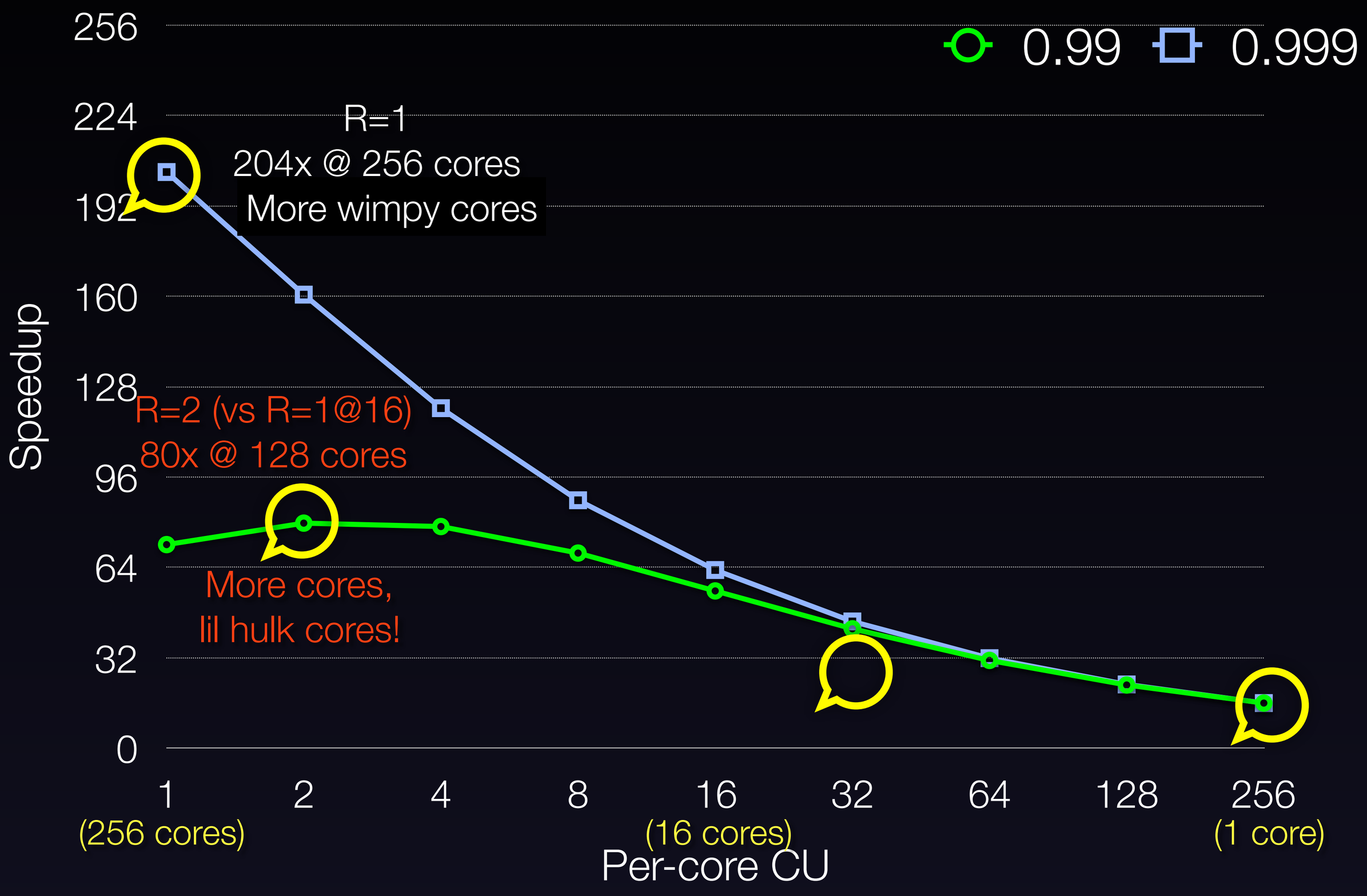
Multicore Moore's Law

- Since 1970s Technology Moore's Law
  - Double transistors every 2 years.
  - Should possibly continue....

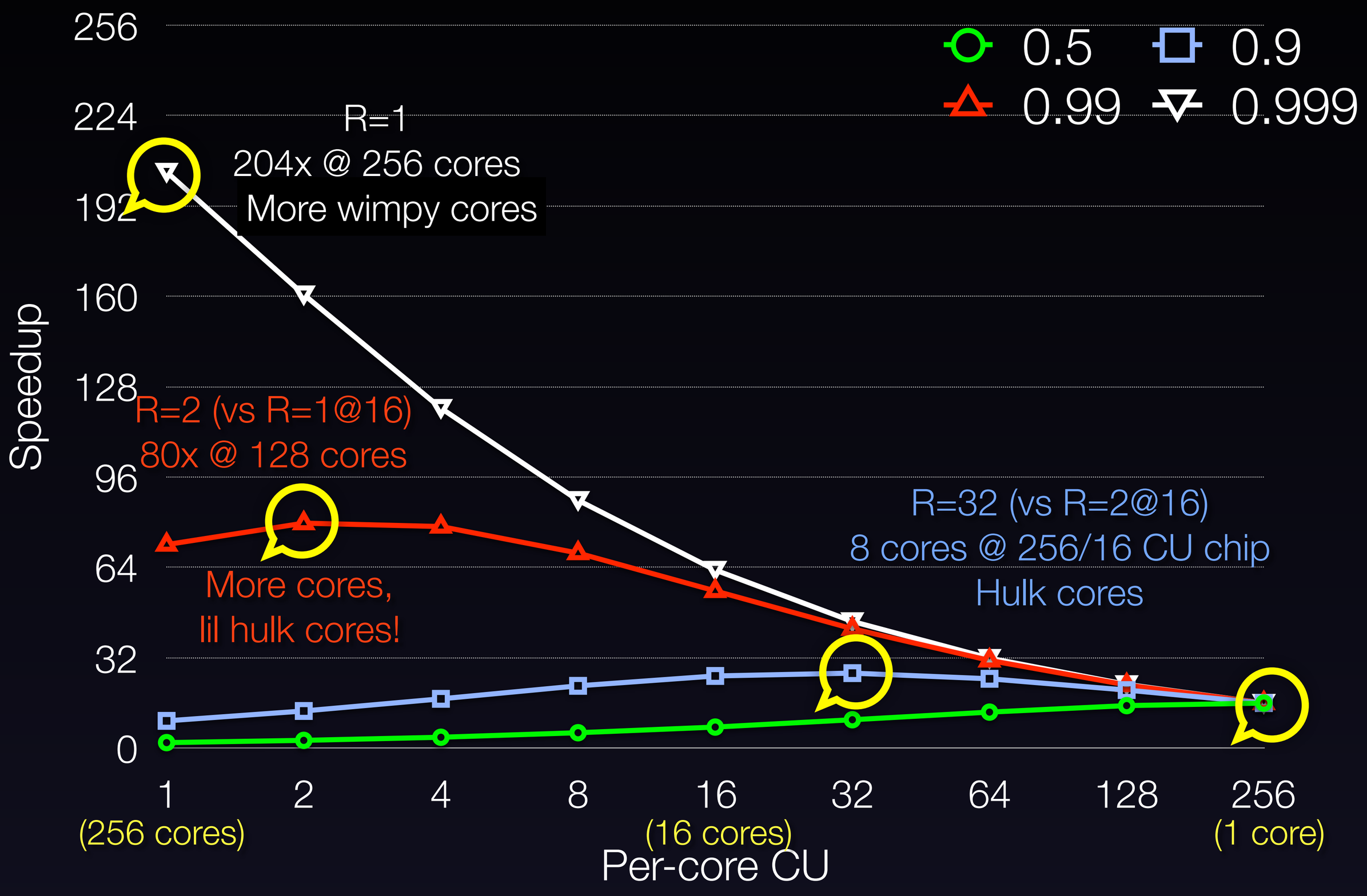
- Microarchitect's Moore's Law
  - double single-thread performance every 2 years
  - Stopped due to power required

  - Multicore's Moore's Law
  - 2x cores every 2 years (1 in 2007- 8 in 2010)
  - Need to double software threads every two years
  - Need HW to enable 2x threads every two years

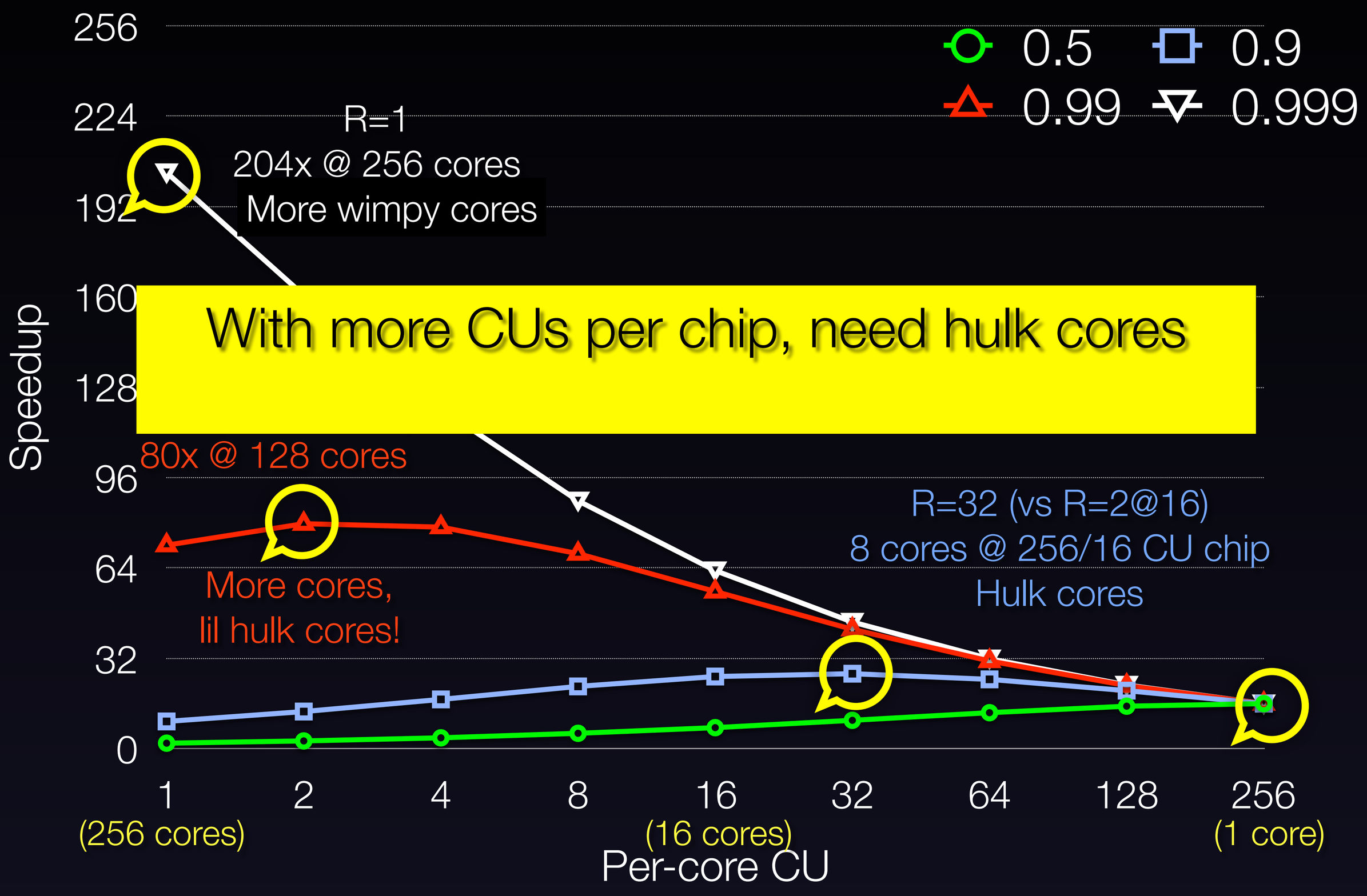
Symmetric Multicore (Chip = 256 CUs)



Symmetric Multicore (Chip = 256 CUs)



Symmetric Multicore (Chip = 256 CUs)



Cost-Effective Multicore Computing

- Is Speedup (N cores) < N that bad ?

- It depends on cost of adding cores.
  - $$$, Power
  - Cost-ratio = Cost (N_{cores}) / Cost (1)

- If chip budget is cost, Cost-ratio << 1.
  - Much of multicore cost outside core [IEEE 1995]
  - Caches, Memory Controller, SSD etc.
- If power is cost, cost-ratio can approach 1

- Multicore computing effective if Cost-ratio > N
  - Intel 6 core = $1600; AMD 10-core 2000$
  - If 10-core speedup >1.25x, then cost-effective

Multicores in Servers and Clients

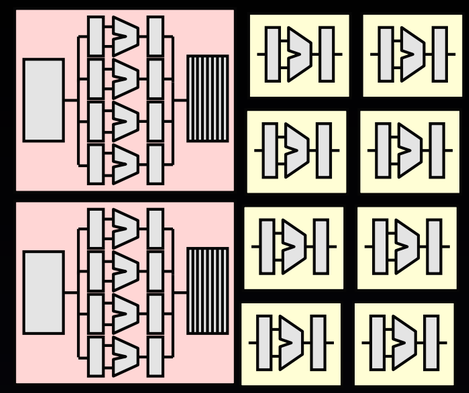
- Multicore parallelism where cost-ratio is low and ap

Causing move to cloud computing

- Cl
  - Smart-phones just moved to dual-cores
  - how many cores?

- Servers
  - can use vast parallelism (Mapreduce, data analysis)
  - natural overlap across clients

Asymmetric Multicores



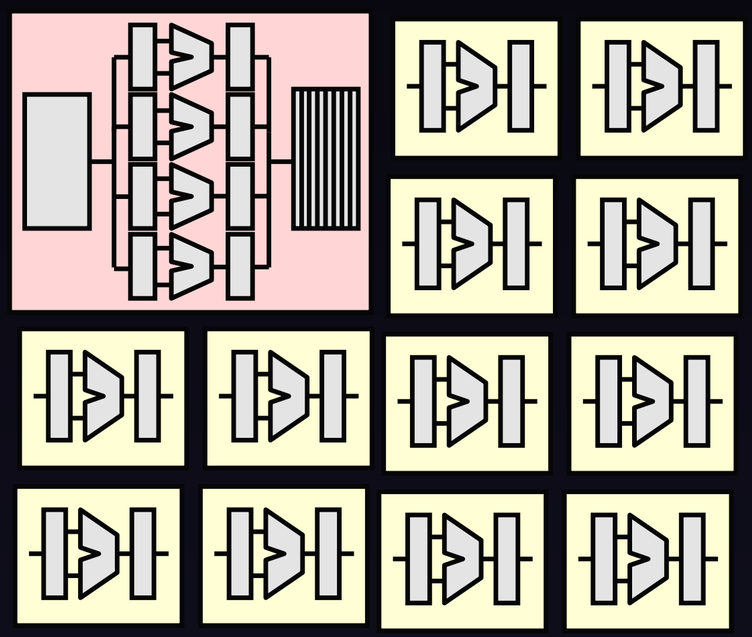
- Enhance some cores to improve performance for serial phase.

- Many designs possible (In this talk, 1 Hulk core)

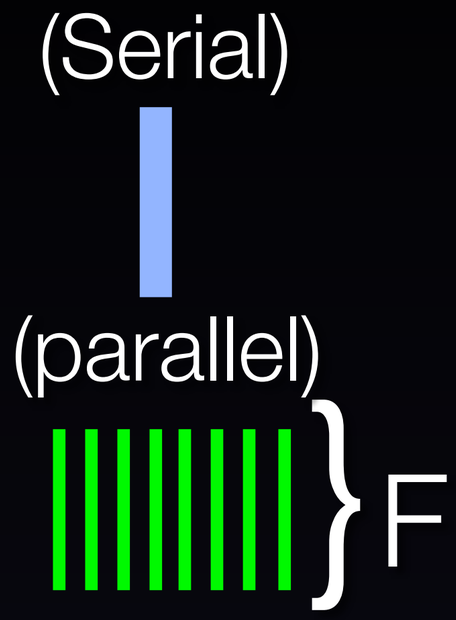
- How to enhance core ?
  - coming up in last 1/3rd of class

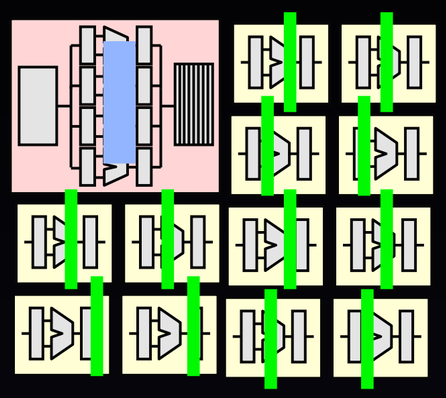
Asymmetric Multicores

- Total chip resources = N CUs
- Assume two-types of cores on-chip
  - One core = R CU, N-R 1 CU cores
  - Total cores = N-R+1



Asymmetric Cores : Performance





Serial Phase = (1-F) / K*Perf (R)

Parallel Phase = (F) / [K*Perf (R) + N-(K*R)]

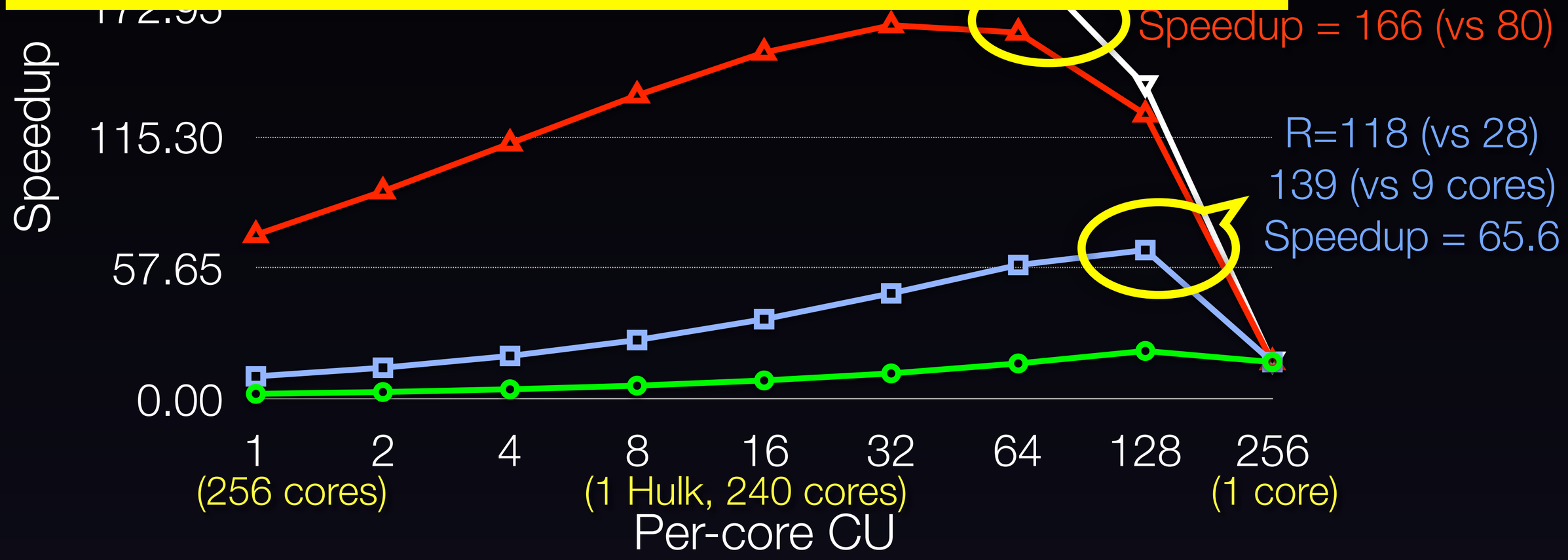
where K is # of Hulk cores.

In our case, K = 1

$$\text{Speedup} = \frac{1}{\frac{1-F}{Perf(R)} + \frac{F}{Perf(R)+N-R}}$$

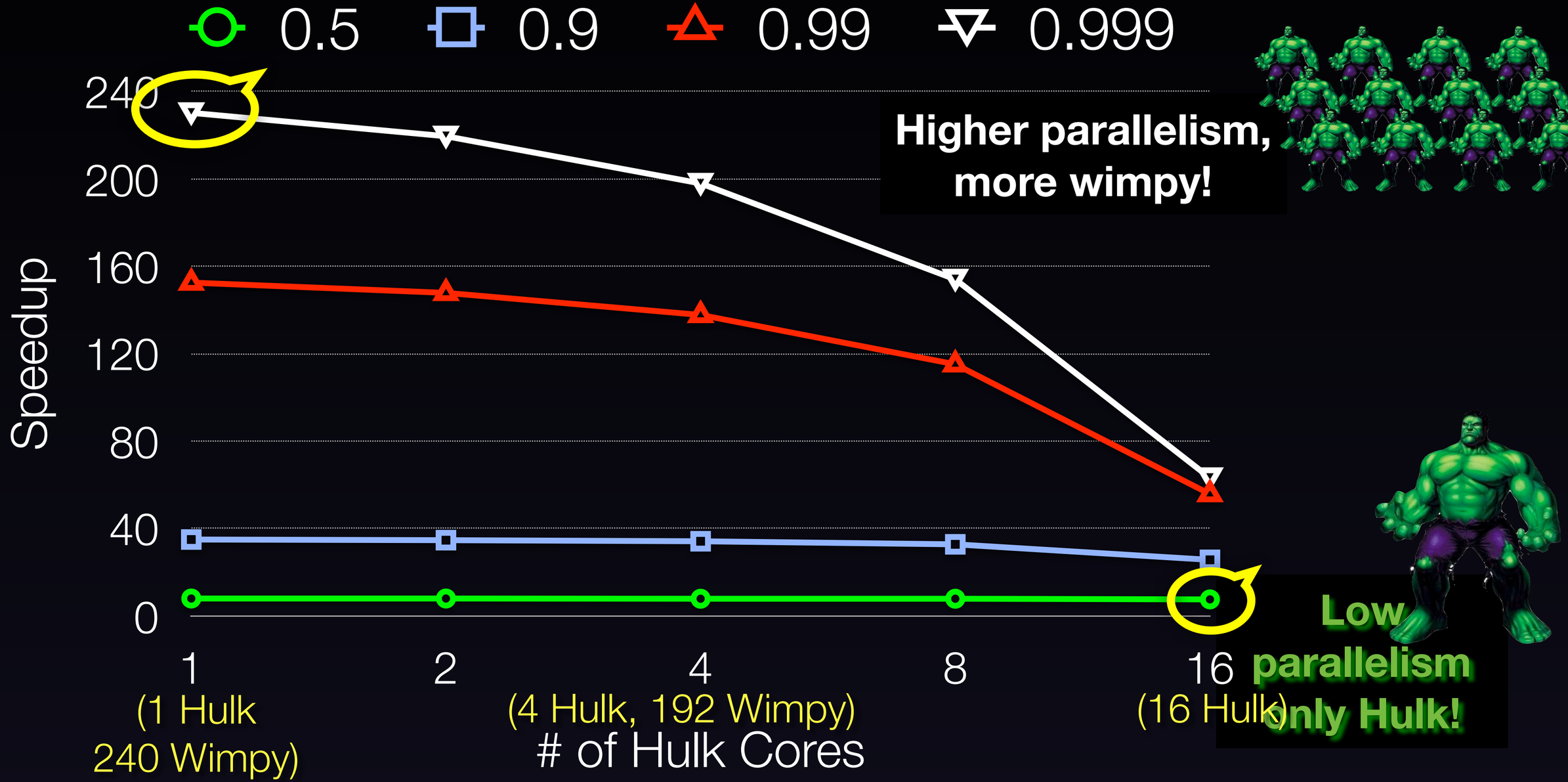
Asymmetric Multicore (Chip = 256 CUs)

Asymmetric cores provide bang for the buck



- Asymmetric cores offer great potential
  - with 1 Hulk core, speedup increases significantly
  - helps take care of Amdahl's law

Asymmetric Multicore (Chip = 256 CUs)



As F increases, always increase wimpy cores!

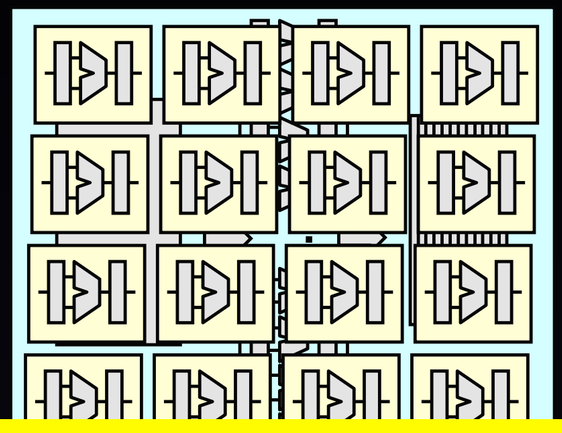
Asymmetric Multicores : Challenge

Task Management :
How to schedule computation?

Locality :
How to keep data close to task?

Coordinate Tasks :
How to synchronize data?

Morphing Multicores



- C
  - Advantage : Can harness all cores on the chip
  - Core optimized
- At runtime glue R 1CU cores to create R CU core
  - improves performance for serial phase
- How to dynamically glue cores ?
  - Not the focus; need’s future research

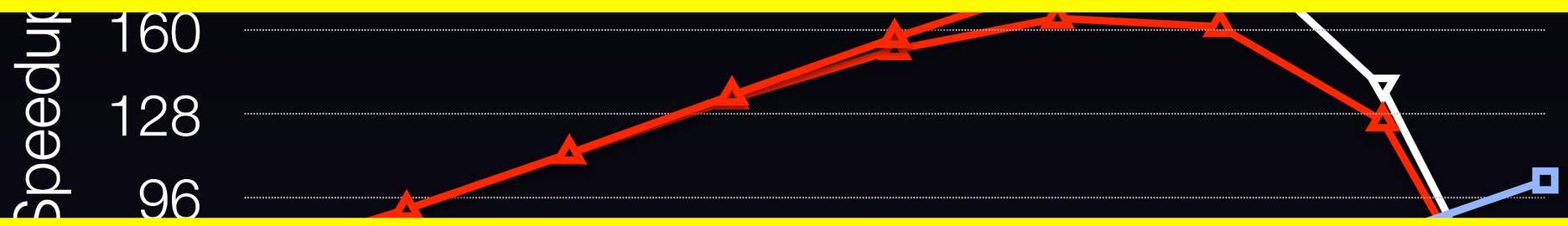
Morphing Multicores : Performance

- N 1CU cores, from which R 1CU cores glued
- Serial phase uses R CU core at Perf (R)
  - execution time = (1-F)/R
- Parallel phases uses N cores
  - execution time = (1-F)/N

$$\text{Speedup} = \frac{1}{\frac{1-F}{Perf(R)} + \frac{F}{N}}$$

Morphing Multicore (Chip = 256 CUs)

Morphing multicores are awesome!
Especially at higher chip resource levels

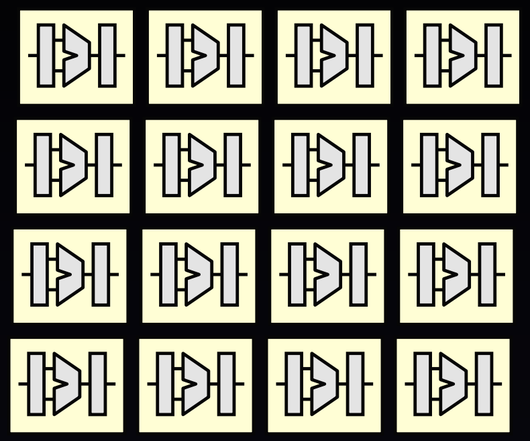


How to glue!

Hulk-Core CU

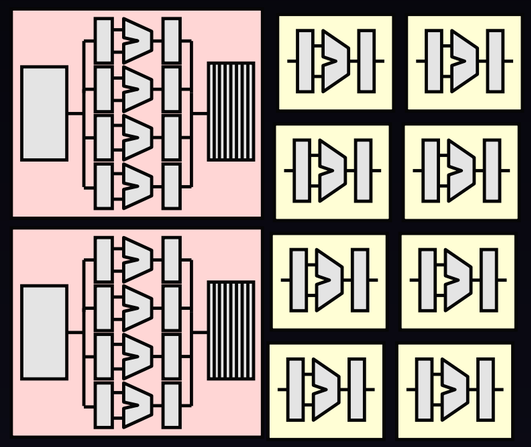
Multicore Amdahl's Law

Symmetric



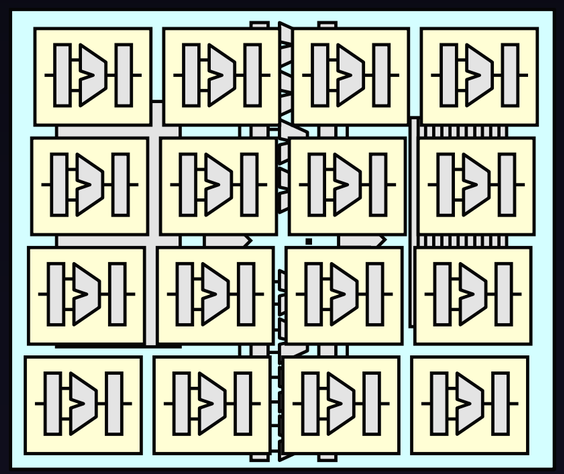
$$\frac{1}{\frac{1-F}{Perf(R)} + \frac{F*R}{Perf(R)*N}}$$

Asymmetric



$$\frac{1}{\frac{1-F}{Perf(R)} + \frac{F}{Perf(R)+N-R}}$$

Morphing



$$\frac{1}{\frac{1-F}{Perf(R)} + \frac{F}{N}}$$

Challenges (1/2)

- Serial Fraction (1-F) has fine-grain parallelism
- Parallel Fraction (F) has serialization overheads
  - You will learn in the next 2-3 weeks.
- Software challenges for asymmetric and dynamic multicores
- How much parallelism in future software?

Challenges (2/2)

Parallelism all the time ?

Amdahl’s Law affects serial fraction ?
Need to increase core speed.

Lots of walls: Power, Area, Shared caches
How to scale CPU performance?