Massively Parallel Algorithms
Organisational Stuff

G. Zachmann
University of Bremen, Germany
cgvr.cs.uni-bremen.de

What You (Hopefully) Get Out of This Course

- Most importantly: *mind set* for thinking about massively parallel algorithms
- Overview of some *fundamental* massively parallel algorithms
- Techniques for massively parallel *visual computing*
- Awareness of the *issues* (and solutions) when using massively parallel architectures
- Programming skills in *CUDA* (the language/compiler/frameworks for programming GPUs)

Is This Course For Me ???

- This course is not for you ...
  - If you don’t like algorithms
  - If you are not ready to do a bit of programming in C
    - The concept of *pointers* should be familiar
  - If you're not open to thinking about computing in completely new ways

Otherwise ...

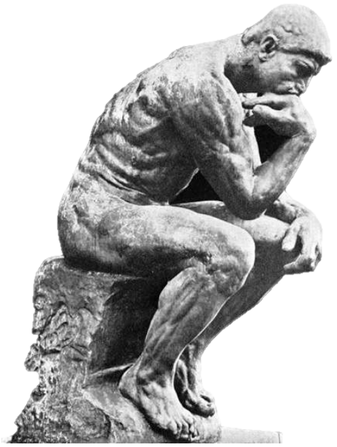
- It will be a richly rewarding experience!

Website

- All important information about this course can be found on:
  http://cgvr.informatik.uni-bremen.de/
  → "Teaching" → "Massively Parallel Algorithms"
- Slides
- Assignments
- Text books, online literature
- Please sign up in StudIP!

Exercises / Assignments

- The two approaches we will pursue in this course:



- Bi-weekly small exercises
  - Recommended: work on them in groups of ~3
  - Skeleton program from us
  - Language CUDA/C++

Grading Criteria of the Exercises

1. "Labeling" variable and function names
2. "Sufficient" comments in body of functions
3. Documentation of functions and their parameters (in/out, pre-/post-condition, what does the function do / not do, ...)
4. Functionality (exercise solved? no bugs? ...)
5. Unless otherwise noted, your code must be parallelized on the GPU

Documentation: minimum

```
// Convert HSV color space to RGB
// input:  h must be in [0,360); s,v must be in [0,1]
// output: r,g,b (out) will be in [0,1]
// Warning: no parameter range checks are done!
__device__
void HSVtoRGB( float *r, float *g, float *b,
               float h, float s, float v )
{
```

Documentation: even better

```
/**  Compute point nearest to q and on an edge of SIG
 *
 * @param q             the current query point
 * @param points        the point cloud
 * @param delaunay      delaunay diagram of the point cloud
 * @param pstar         NN of q (out)
 * @param pstar2        neighbor of pstar (in SIG) (out)
 * @param phat          point closest to q on edge (pstar,pstar2) (out)
 * @param d             distance between q and phat (out)
 *
 * @warning
 *   Assumes that a SIG has been computed!!
 * @bug
 *   Bis jetzt ist pstar2 nur NN zu pstar im Delaunay-Graph, nicht im SIG!!
 **/

void nearest_on_graph( const FPoint & q, const std::vector<FPoint> & points,
                       const Proximity & proximity,
                       unsigned int * const pstar, unsigned int * const pstar2,
                       FPoint * const phat, float * const d )
{
```

The SDK, Needed for Working at Home

- IDE (obviously) of your choice
  - Can be as simple as an ASCII editor and compiler on command line
- CUDA for your platform:
  https://developer.nvidia.com/cuda-downloads
  - Works, of course, only with NVidia graphics cards
    - Anyone interested in trying out ZLUDA? (https://github.com/vosen/ZLUDA)
    - Or AMD's translation tool HIPIFY?
  - If your laptop/desktop does not contain NVidia, use the pool or our lab
  - MZH 0245 (MIR, ground floor), Tuesdays 10-14h and Thursdays 10-14h
  -

Overall Grades & Examinations

- You have two options:

  1. *Regular oral exam,* ca. ½ hour per student
  2. *Mini oral exam* (so-called "Fachgespräch"), ca. 10 minutes per student

- The formula for calculation of your grade with option 2:
  - Assignments ⟶ grade A
    - 95% of all points ⟶ A = 1.0
    - 40% of all points ⟶ A = 4.0
  - Mini oral exam ⟶ grade B
  - Overall grade $= \min\left\{\frac{1}{2}\cdot(A+B),\ B\right\}$ ("min" means "better of the two")
    - Under the condition: A ≥ 4.0 && B ≥ 4.0 !

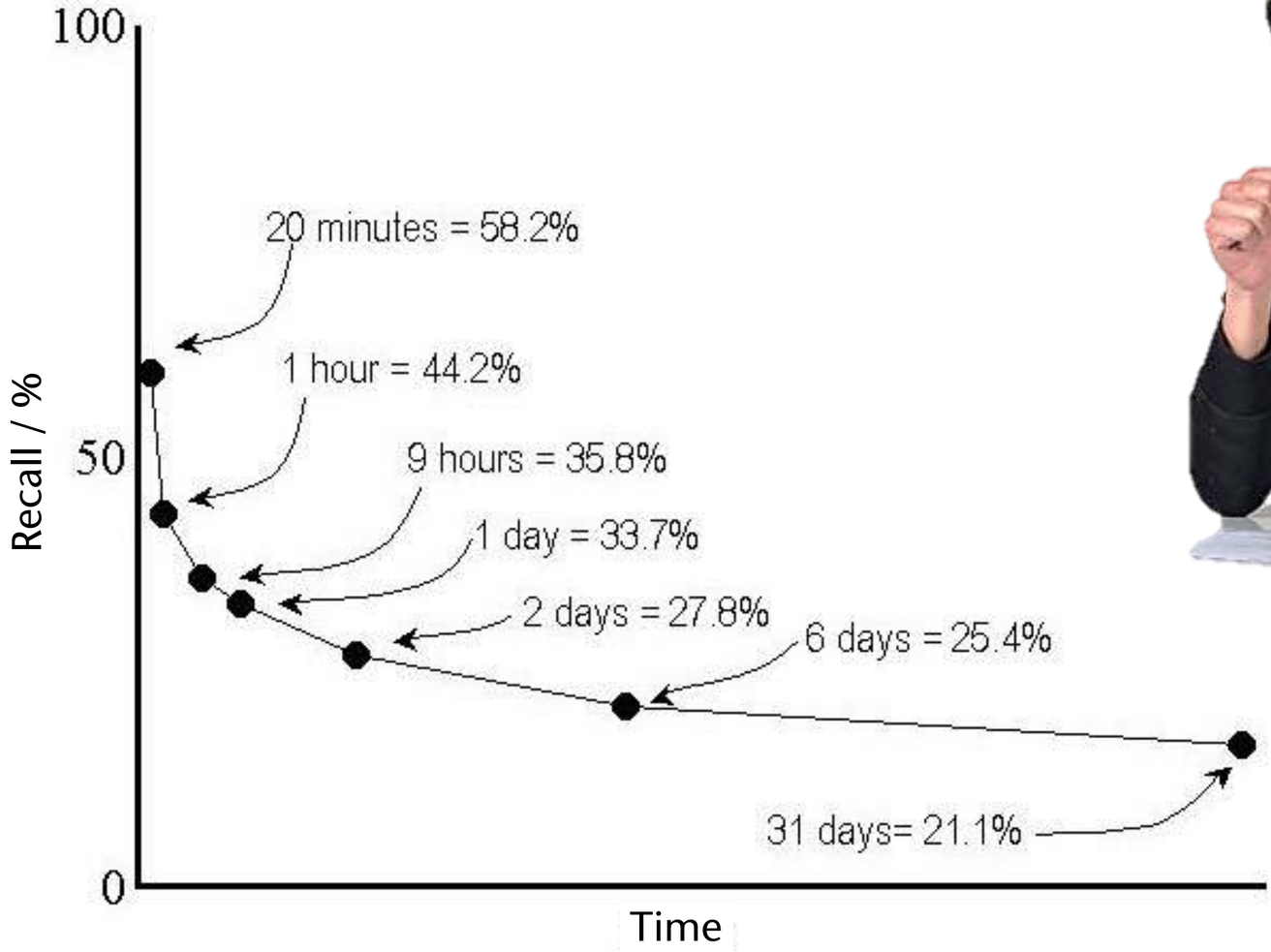
- Note: in both cases, *all of the material could be topics for the exam*!

A Quote

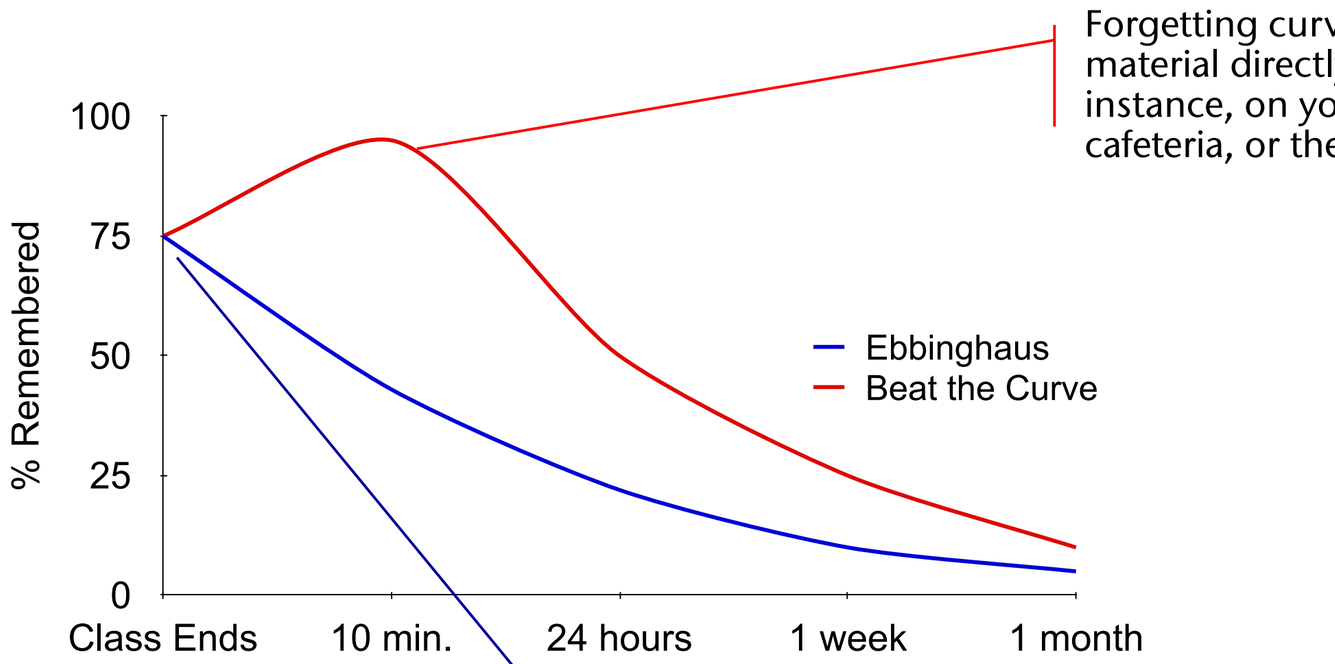
I hear and I forget.
I see and I remember.
I do and I understand.

[Attributed to Confucius]

The Forgetting Curve (Ebbinghaus)



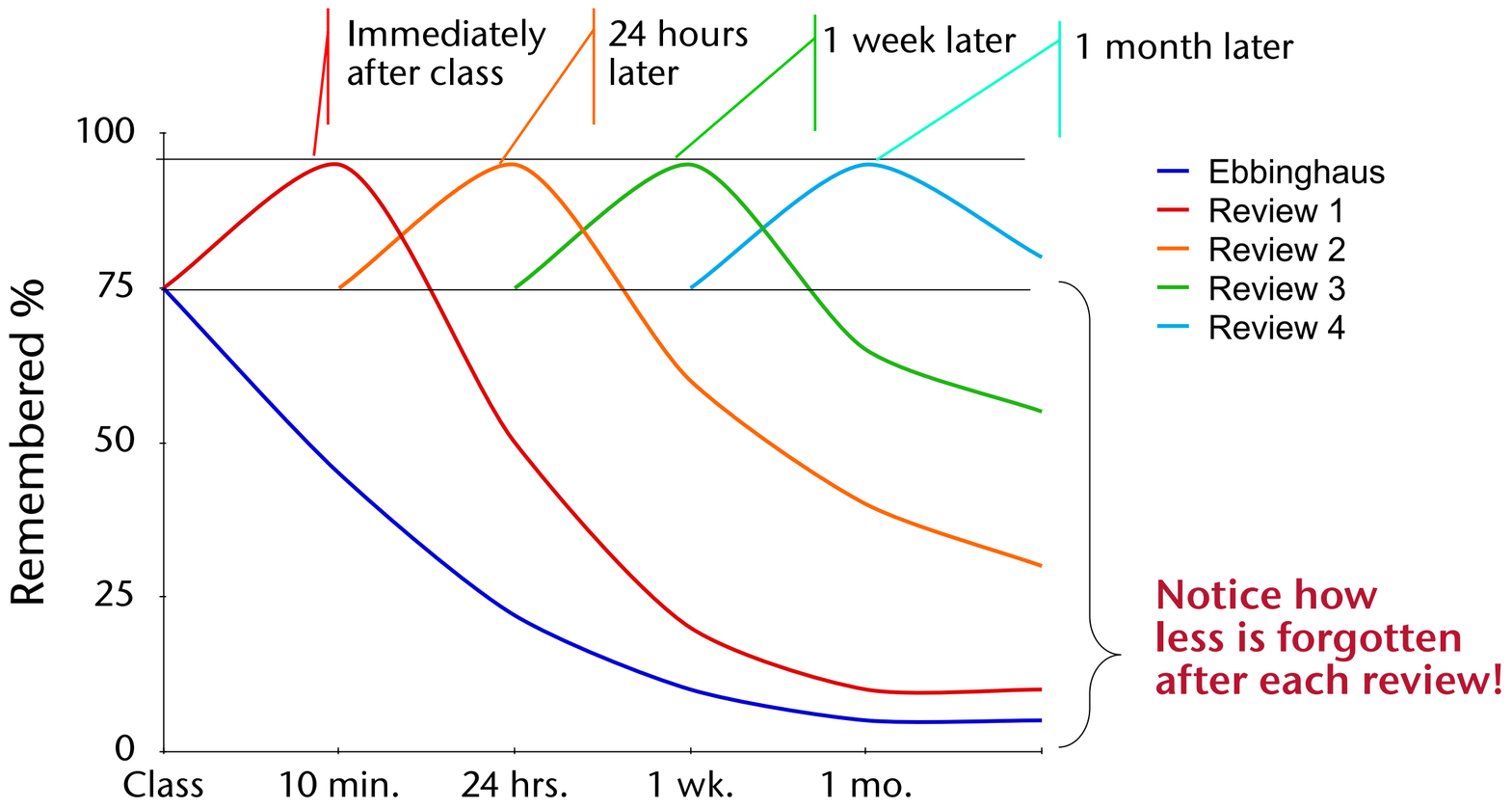
Beating the Forgetting Curve



Forgetting curve when you recap the material directly after class (for instance, on your way to the cafeteria, or the tram station)

Forgetting curve actually starts here as we typically remember only about 75% at the end of a lecture – so we have less to remember

Overcoming the Curve



Average Retention Rates

- Just listening 5%
- Reading 10%
- Audio Visual 20%
- Demonstration 30%
- Discussion 50%
- Practice by doing 75%
- Teach others 90%