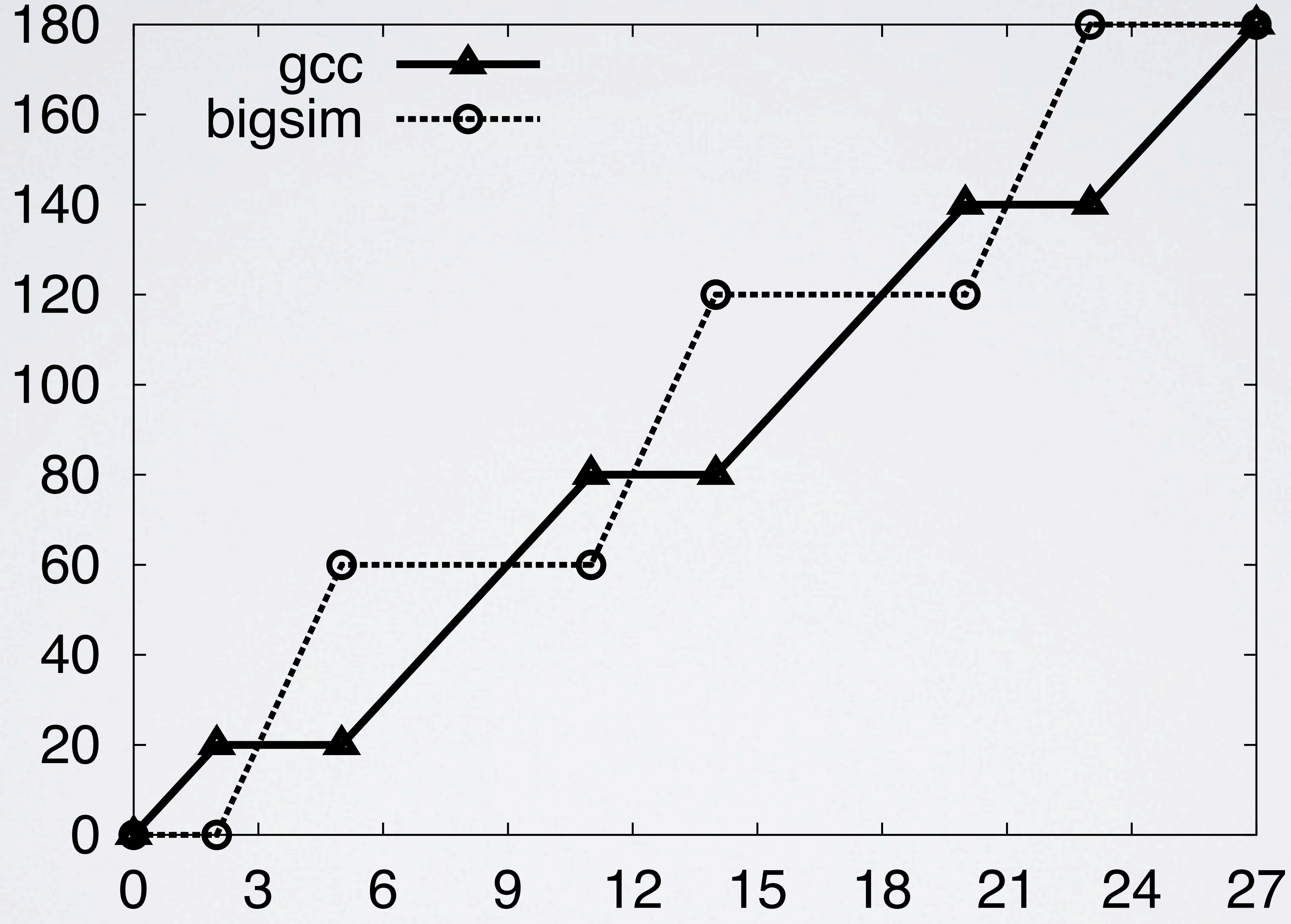
BORROWED VIRTUAL TIME

Kenneth J. Duda & David R. Cheriton

MOTIVATION

- general purpose schedulers provide only **fair sharing**
- degrades latency-sensitive applications
- specialized **real-time** schedulers require specification
- forces applications into task model
- find a middle ground

WEIGHTED FAIR SHARING



VIRTUAL TIME

- each thread carries a **virtual timestamp**
- increases when the thread runs
- increment inversely proportional to thread's **weight**
- waking from sleep advances virtual time to the minimum of all runnable threads
- switch to thread with smallest virtual time when running thread exceeds **lead bound**

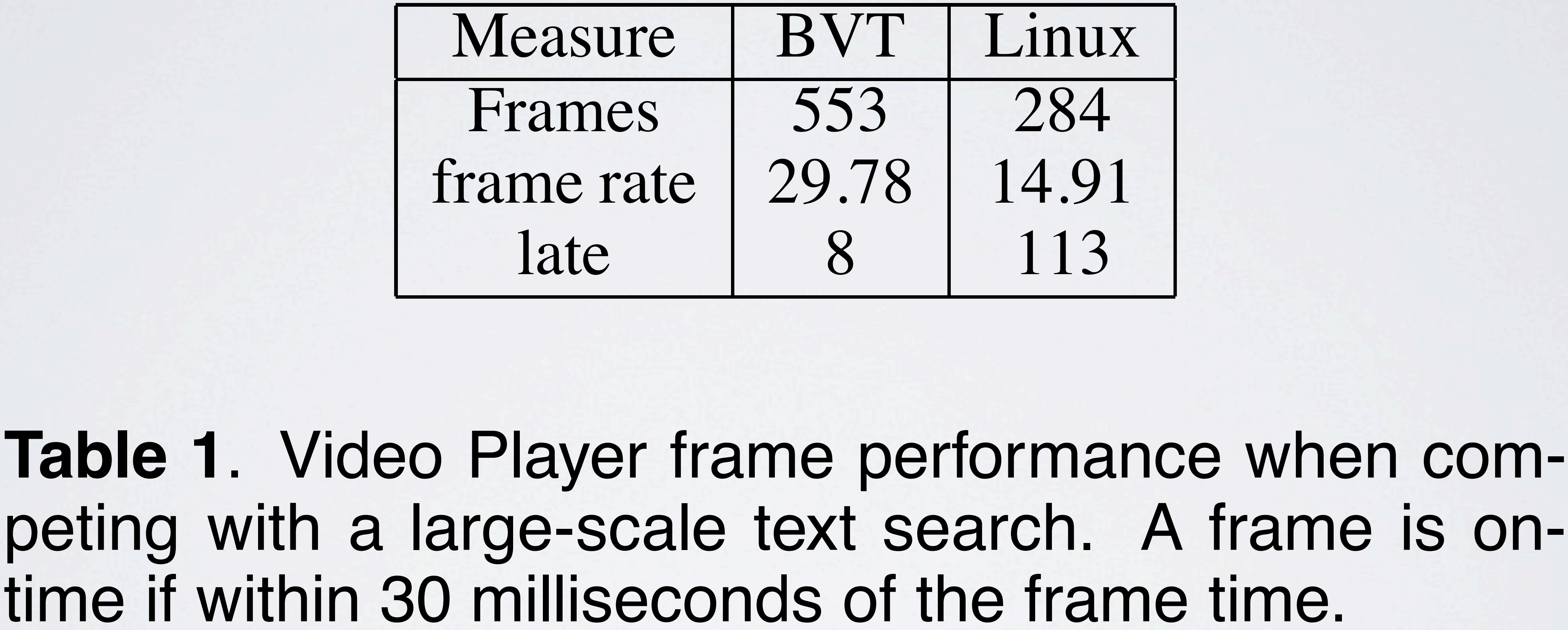
DISPATCH LATENCY

- threads can **warp** back in time
- effective virtual time = actual virtual time – warp time
- effective virtual time is used for scheduling
- allows a thread to **borrow** time from its future execution
- warping is constrained by **warp time limit** and **unwarp time requirement**

THOUGHT EXPERIMENT

- write a video player with this concept
- you have to decide on the following parameters:
  - weight: CPU share you need
  - warp: global dispatch priority
  - limits: how nice you are to others

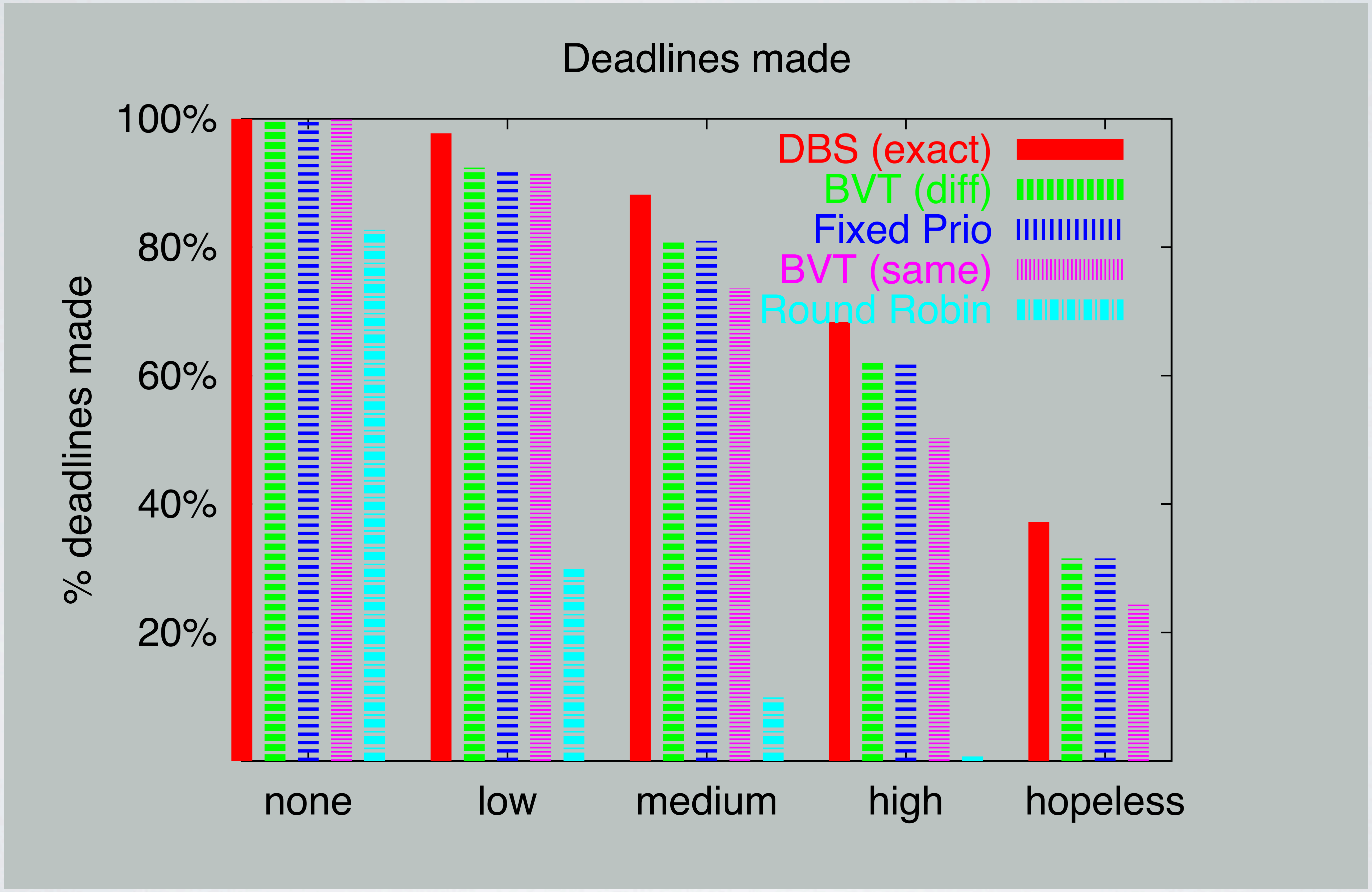
EXPERIMENTS



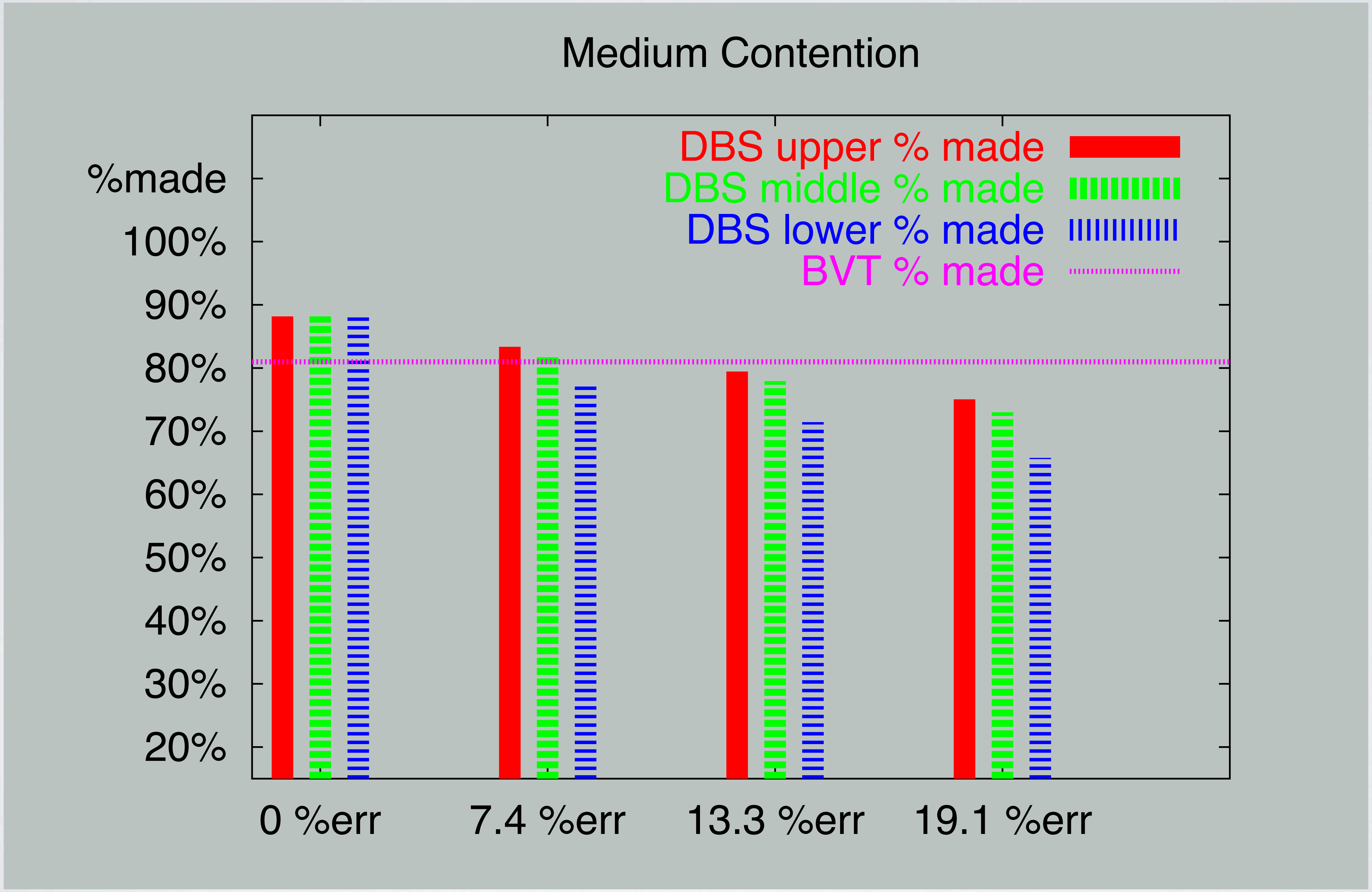
Measure	BVT	Linux
Frames	553	284
frame rate	29.78	14.91
late	8	113

Table 1. Video Player frame performance when competing with a large-scale text search. A frame is on-time if within 30 milliseconds of the frame time.

DEADLINE SCHEDULING



DEADLINE SCHEDULING



SCENARIOS

- hard real-time: relative CPU shares become absolute rates when you run **admission**
- pick warp values **like priorities**
- **two-level** BVT scheduling:
  fully nested, warp threshold, direct

CONCLUSION

- BVT is great
- simple mechanism
- generally applicable
- efficient
- outperforms EDF

- BVT’s contribution is small
- unintuitive parameters
- with admission (not included)
- … OK, maybe
- if compared unfairly

Subtracting a warp factor from a task's timestamp seems to be like saying, *do this yesterday*—it has no coherent meaning. Instead, BVT uses virtual time as a simple mechanism for ordering tasks: warping a task moves it up in the ready queue, and this reduces its dispatch latency. As a result, it is not clear exactly what kinds of behaviors BVT can provide. For instance, how do multiple warped tasks interact with each other? How does a user set the various warp parameters for all applications in order to produce a desired overall system behavior?

DISCUSSION

- How useful are fair-share schedulers to applications?
- Is deadline not a more natural way to specify timing requirements?
- Is this whole fairness-thing a leftover from the bygone days of multiuser terminal servers?
- fairness first, timing second vs. timing first, fairness second