Resource Scheduling in Enhanced Pay-Per-View (EPPV) Continuous Media Databases

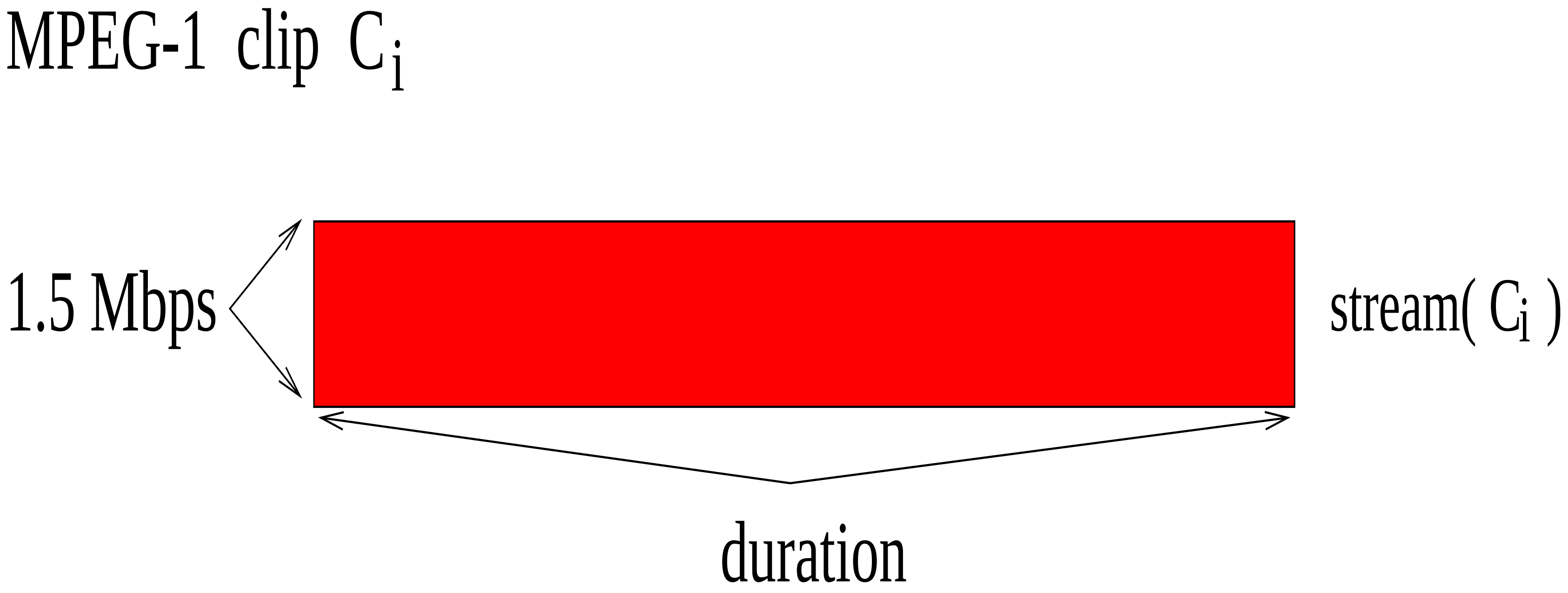
Minos N. Garofalakis *(Univ. of Wisconsin – Madison)*
Banu Özden *(Bell Laboratories)*
Avi Silberschatz *(Bell Laboratories)*

Outline

- Introduction and Motivation
- Problem Formulation and System Model
- Clustering
- Striping
- *Main Contribution:* Scheduling Tree Structure & Algorithms
- Experimental Results
- Conclusions and Future Work

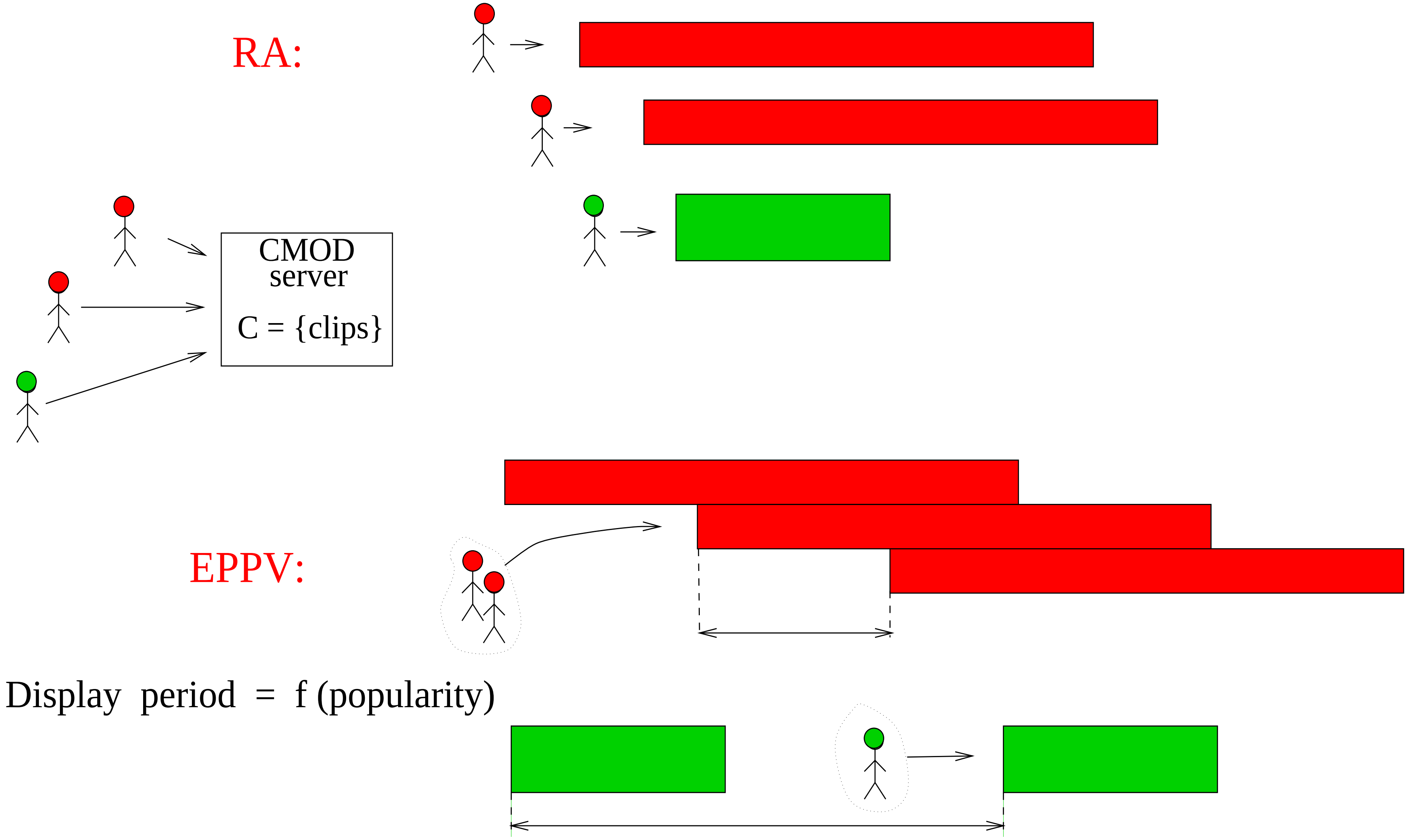
Introduction and Motivation

- Continuous Media: Real-time resource requirements for delivery



- Objective: Maximize number of concurrent clients given bounded resources

Service Models: Random Access vs. EPPV



Introduction and Motivation (cont.)

Why EPPV?

- Cost-effective: concurrency not limited by available resources
- Predictable response times

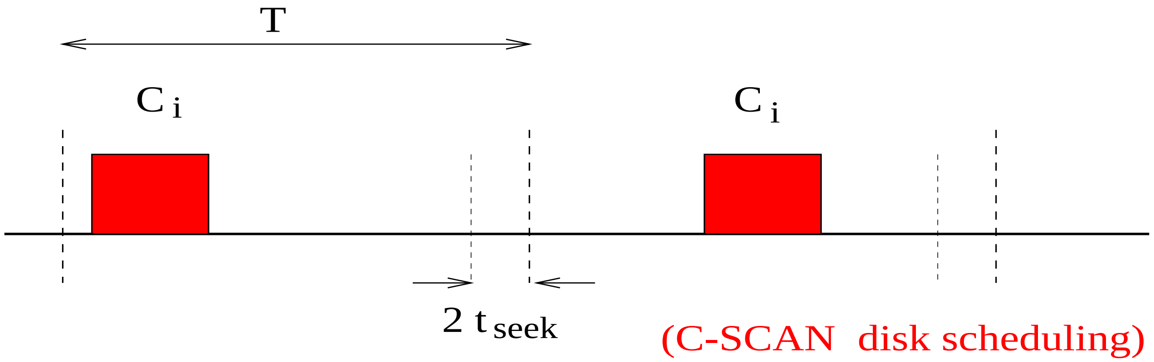
Challenge: Complex periodic task scheduling problems

The Problem:

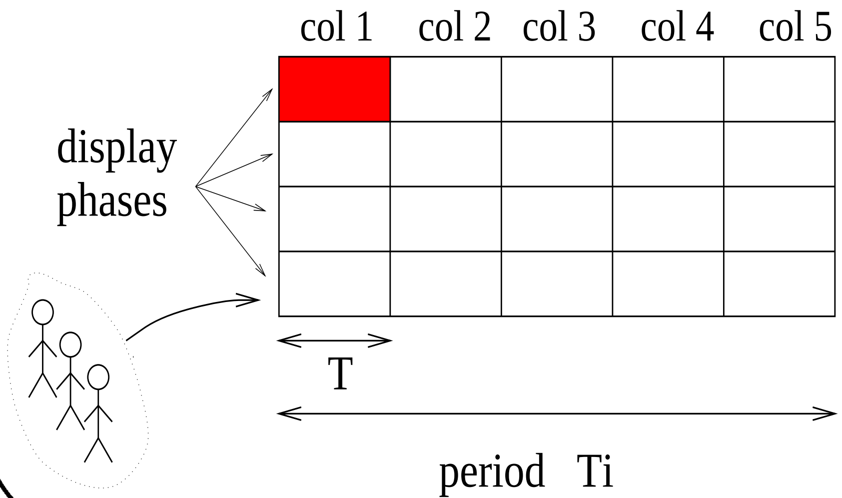
- **Given:** disk H/W parameters, data layout strategy, and set of clips C
- **Find:** placement and feasible schedule for subset of C that maximizes a "benefit" (e.g., scheduled disk bandwidth)

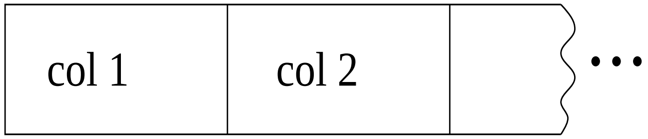
System Model

- Round-based Retrieval: data for streams retrieved in rounds of length T



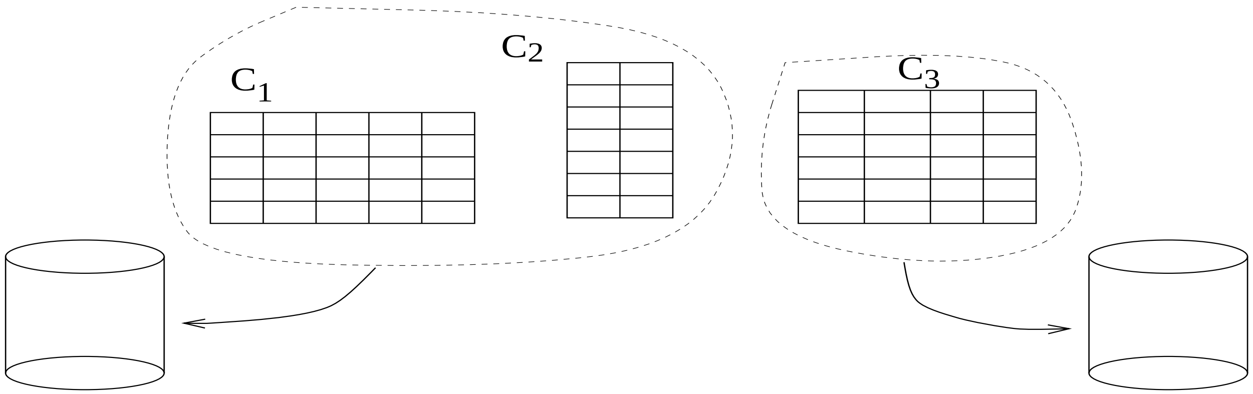
- Matrix-based Allocation [Özden et al., VLDB'94]: designed for EPPV





EPPV under Clustering

- Clustering: entire clips mapped to individual disks



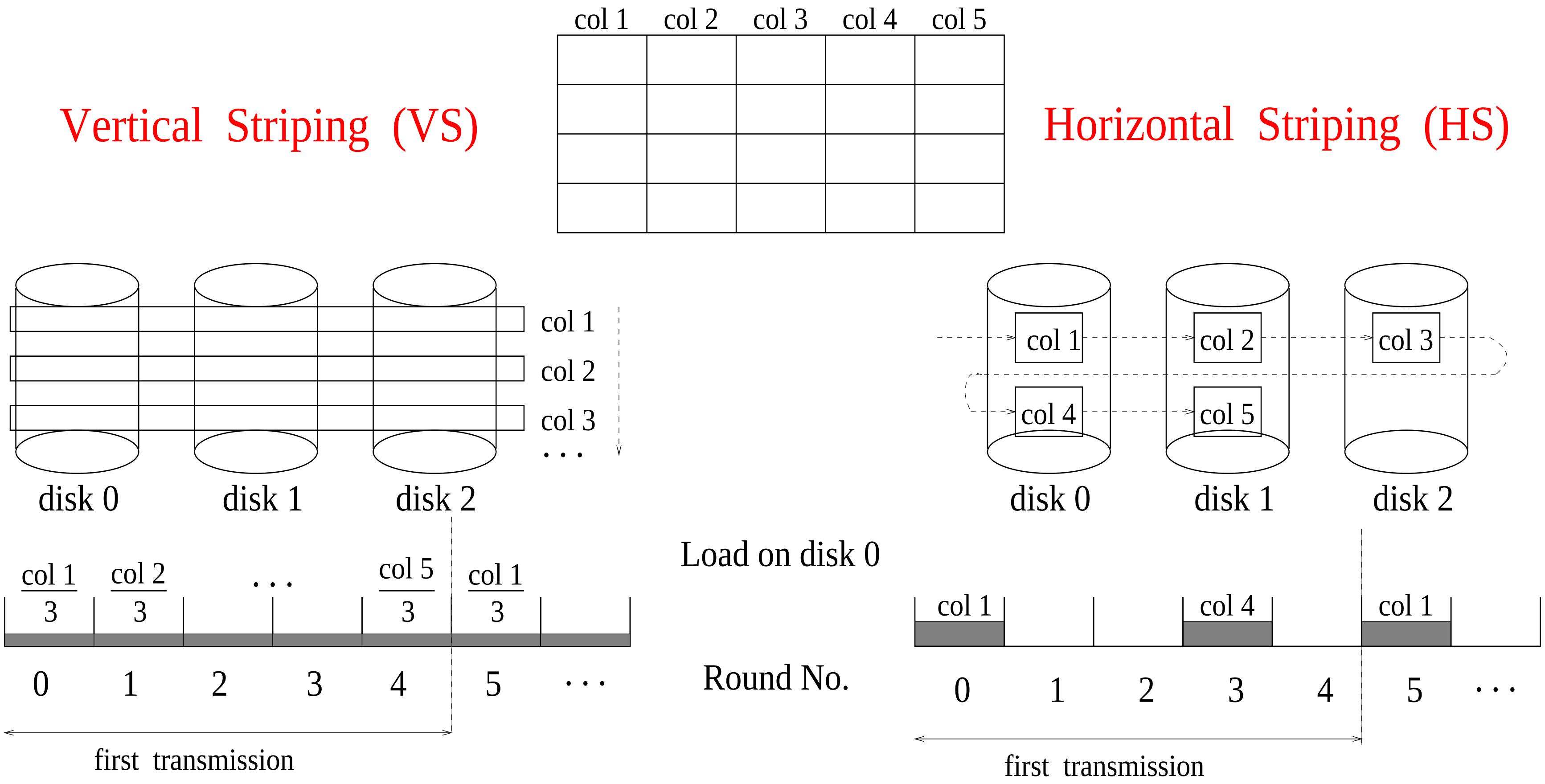
- size(C_i) $=$ 2-dimensional: fraction of disk bandwidth and fraction of disk storage consumed by C_i

  value(C_i) $=$ disk bandwidth effectively utilized by C_i

- NP-hard "knapsack" problem

- Near-optimal heuristic solution (PackClips)

EPPV under Striping



- VS: limited scalability because of latency overheads
- HS: "composite" periodic tasks (period $n_i \cdot T$, subtasks $n_{disk} \cdot T$ apart)

HS and Periodic Maintenance

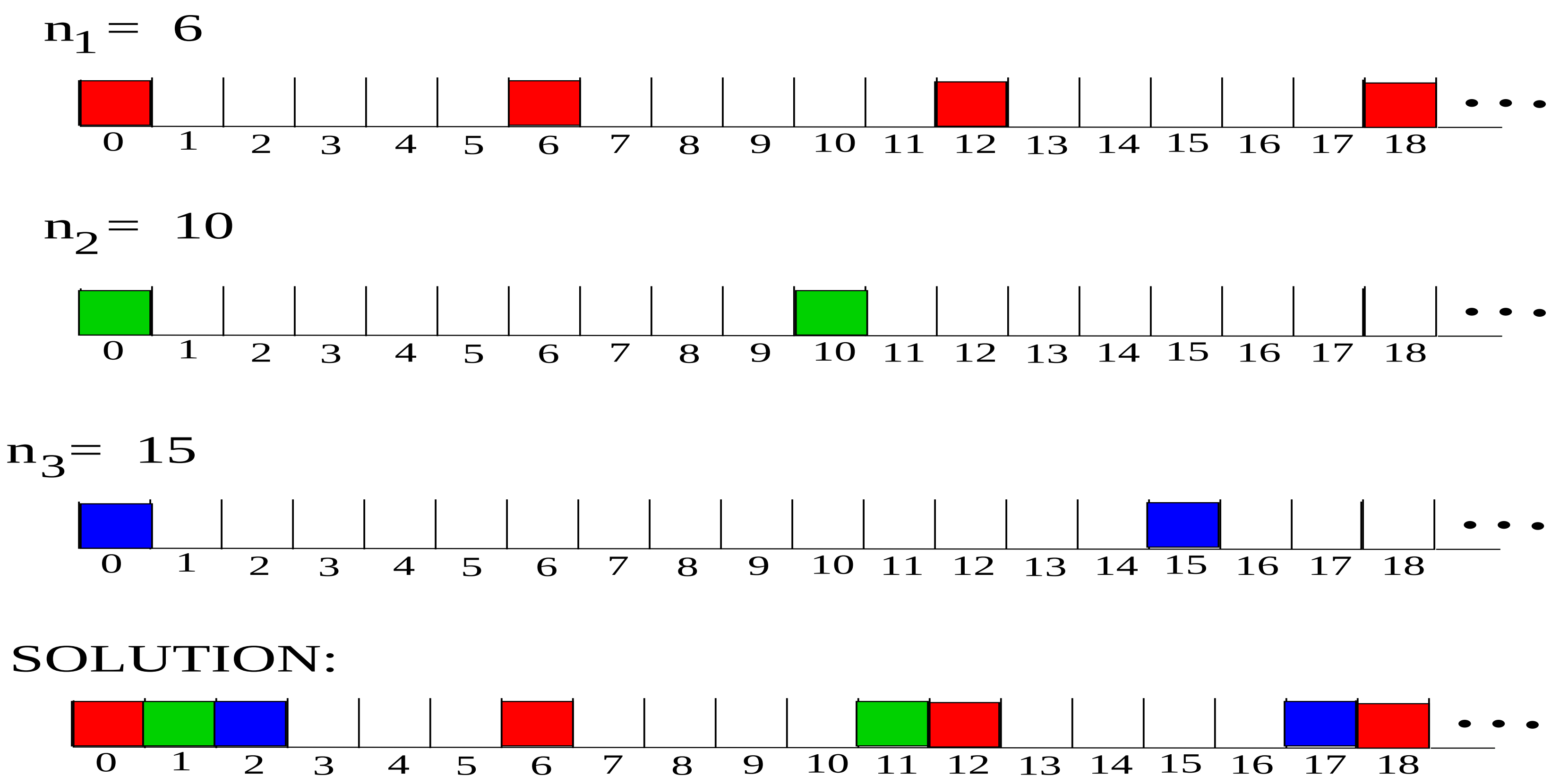
Collision-free scheduling for HS

- Special cases: conditions based on *Generalized Chinese Remainder Thm*
- General case: generalization of *Periodic Maintenance*

Periodic Maintenance: Given periodic tasks $\{C_i\}$ with periods $\{n_i\}$, is there a mapping onto integer time slots s.t. successive occurrences of C_i are exactly n_i apart and no collisions occur?

- Special case of HS scheduling (one subtask per task)
- NP-complete in the strong sense [Baruah et al., 1990]

Periodic Maintenance: Example



- No solution exists, e.g. for $\{3, 5\}$ (*Chinese Remainder Theorem*)

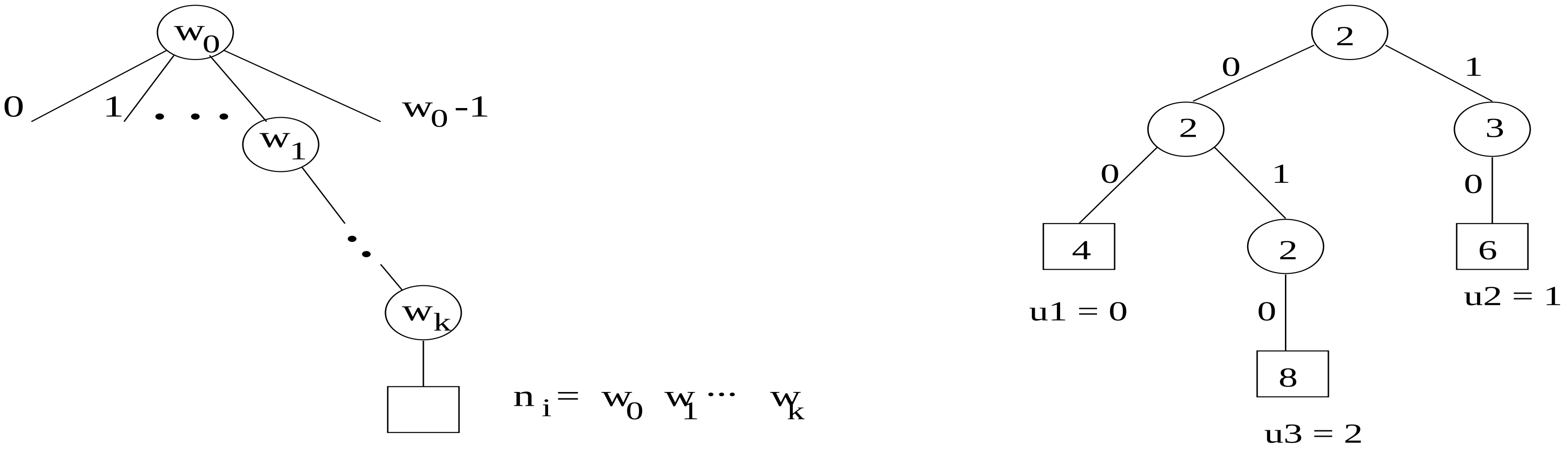
Our Main Contribution: Scheduling Trees

Goal: provide solutions for Periodic Maintenance; extend to HS

Scheduling Tree: integer weighted nodes and edges, where

1. an internal node with weight w has $\leq w$ edges with distinct weights in $\{0, \ldots, w-1\}$

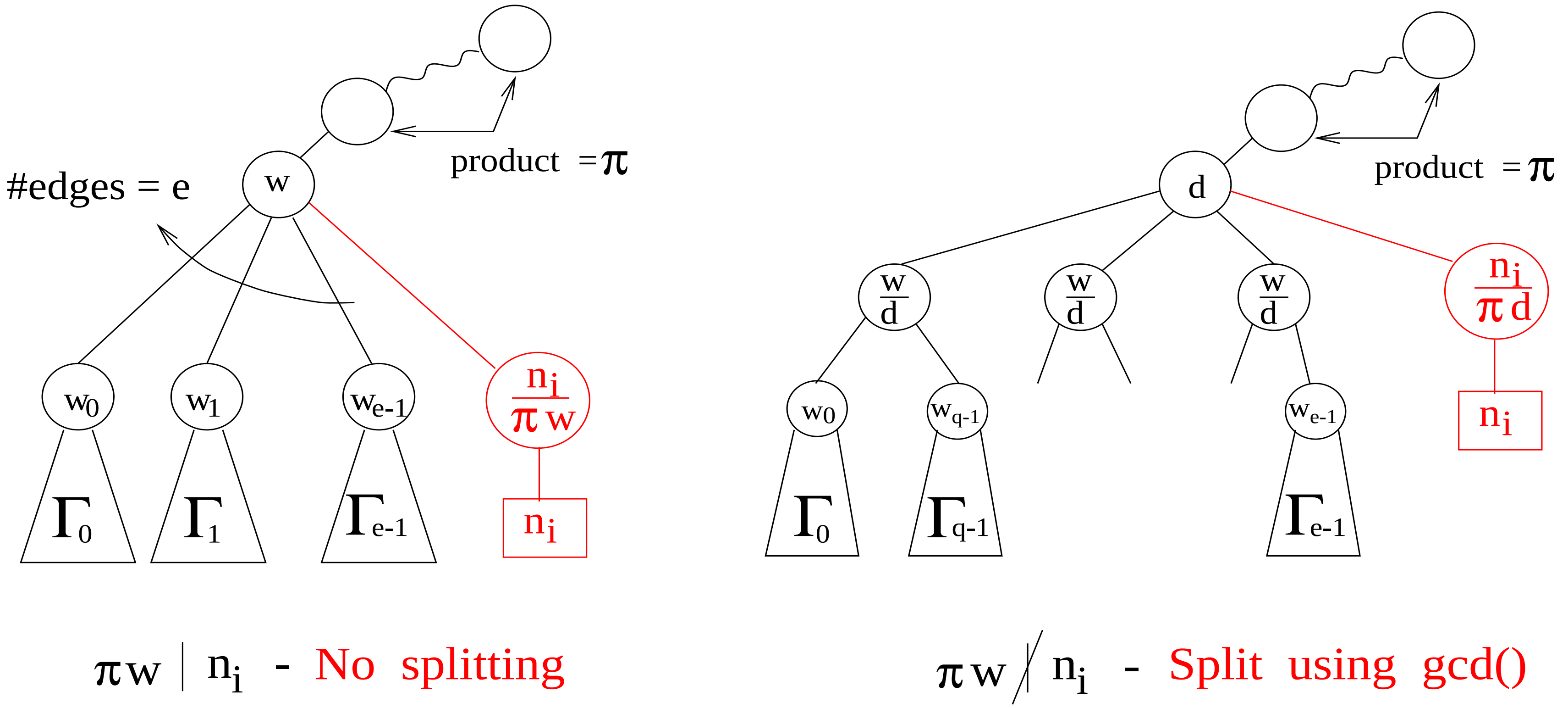
2. a leaf node corresponds to period n_i s.t. n_i = product of all ancestors



- $\exists$ Scheduling Tree is only sufficient for Periodic Maintenance; e.g., $\{6, 10, 15\}$

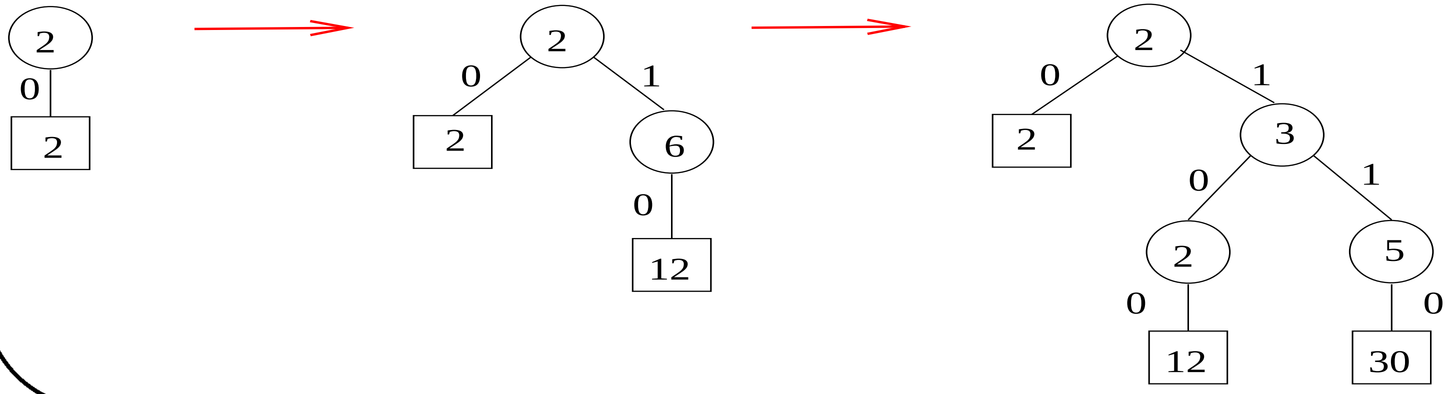
BuildTree: A Heuristic for Building Scheduling Trees

- Based on *candidate nodes*
- Node is candidate for n_i iff $\gcd(w, \frac{n_i}{\pi}) \geq \frac{w}{w-e}$



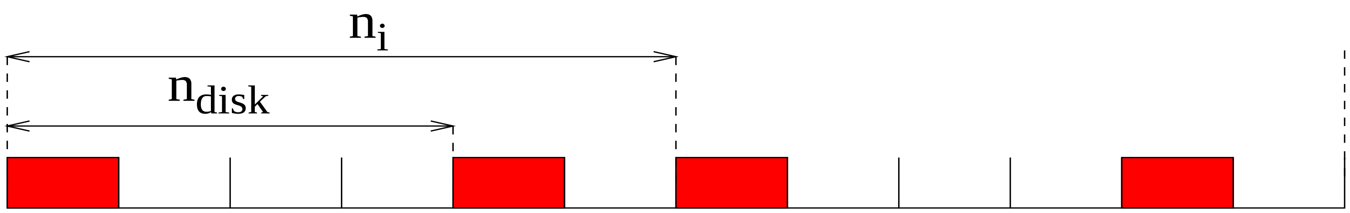
BuildTree: A Heuristic for Building Scheduling Trees

- Main Ideas:
  - Maintain candidates incrementally as tree is built (efficient...)
  - Consider tasks/periods in decreasing order of benefit
  - Place under candidate that minimizes total loss over all remaining periods
- Complexity = $O(N^3)$
- Simple example: Period list = < 2 ,12 , 30 $>$



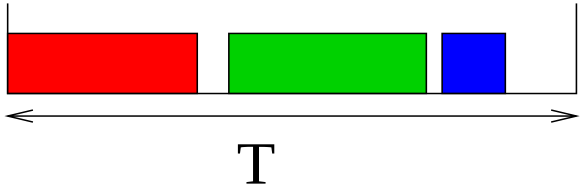
Scheduling Tree Solutions for HS

Equidistant subtasks?



- Each subtask is itself periodic with period n_i
- Splitting cannot distort relative placement
- Ensure all subtasks are scheduled together at regular intervals of n_{disk}

Scheduling Tree Solutions for HS (cont.)



Multi-task slots?

- Leaf nodes/slots with sufficient capacity can also be candidates
- Combining ("packing") independently built trees to improve slot utilization

Combining Scheduling Trees

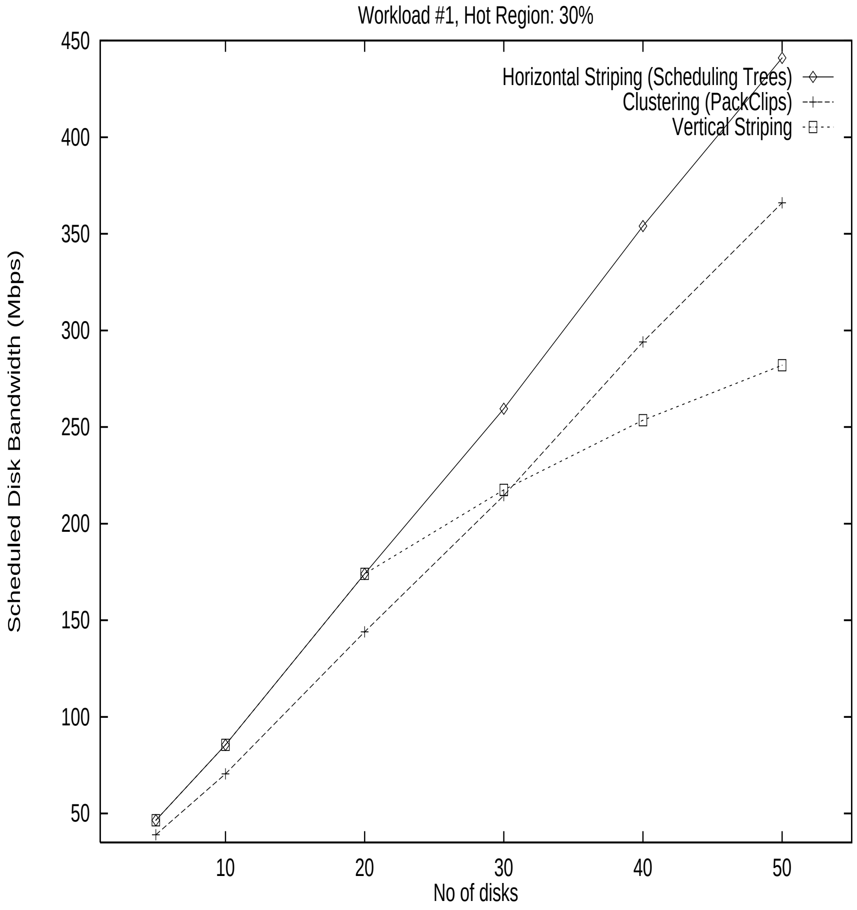
- *Worst-Case Collision Assumption*:
  - size(tree) = max. load among all leaves
  - value(tree) = total benefit of all tasks in tree
- Knapsack problem: use PackClips to approximate solution
- More sophisticated approaches: use Chinese Remainder Theorems

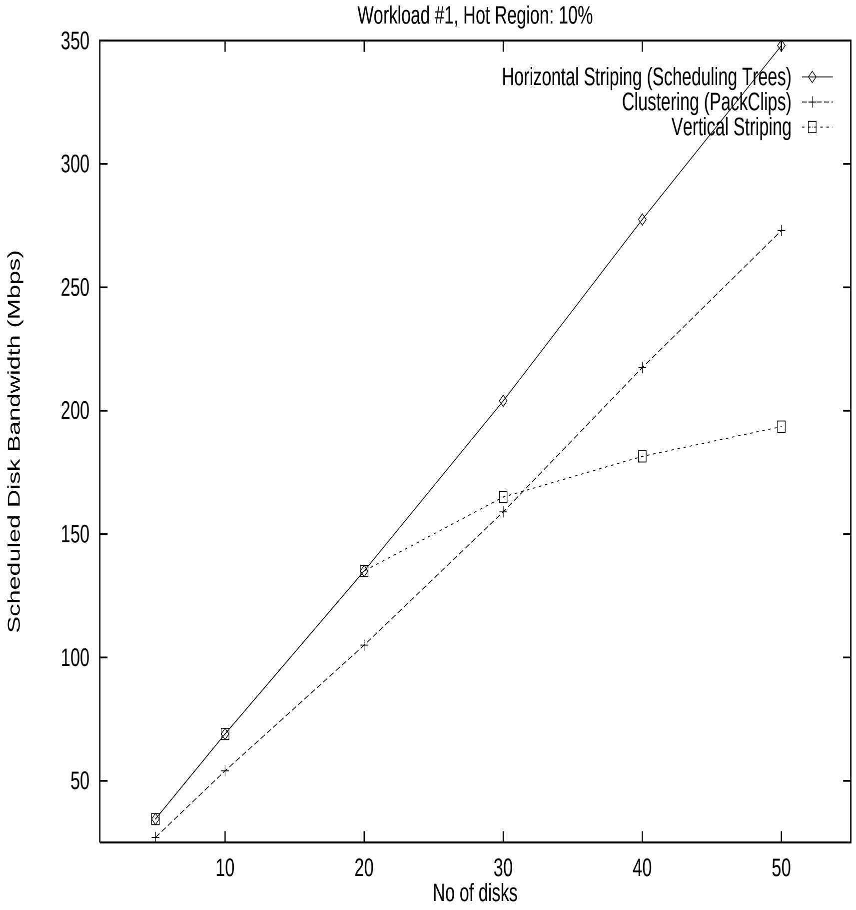
Experimental Results

- Compare EPPV scheduling schemes under Clustering, VS, and HS
- Metric: effectively scheduled disk bandwidth
- Two workload components
  - Long (90-120min) MPEG-1 movies (1.5Mbps)
  - Short (2-10min) video clips (2-4Mbps)
- "Hot" and "Cold" regions (short/long periods)
- Storage requirements ≈ storage capacity of disks

Experimental Results (cont.)

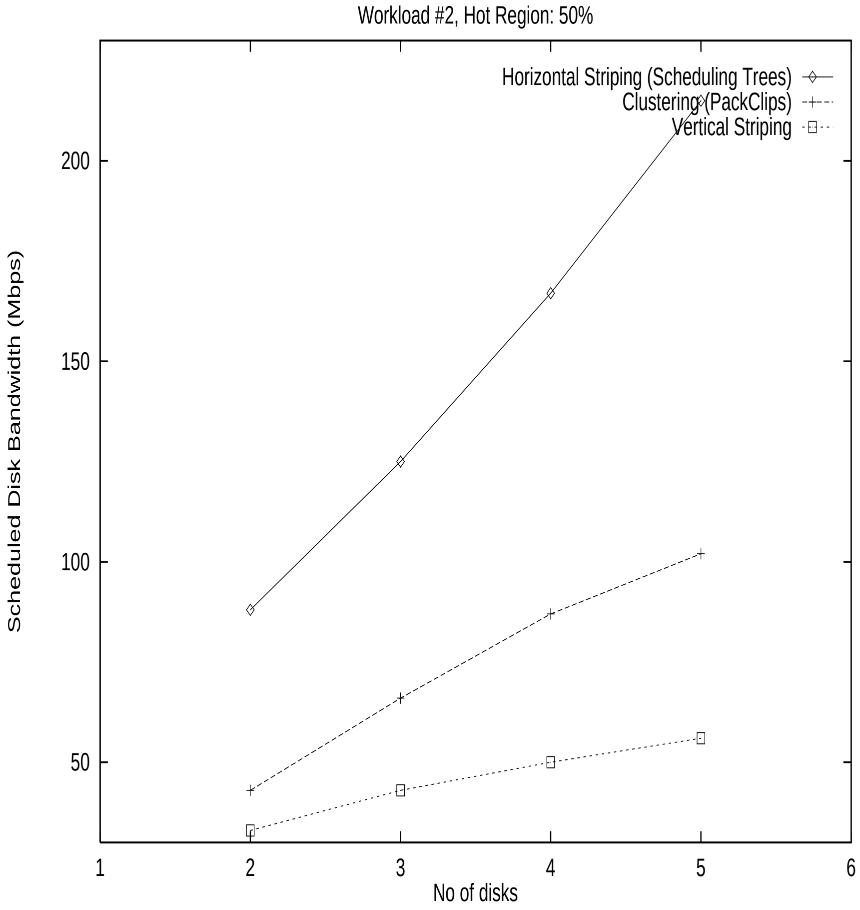
- Long Clips, 30% "Hot"
- Long Clips, 10% "Hot"



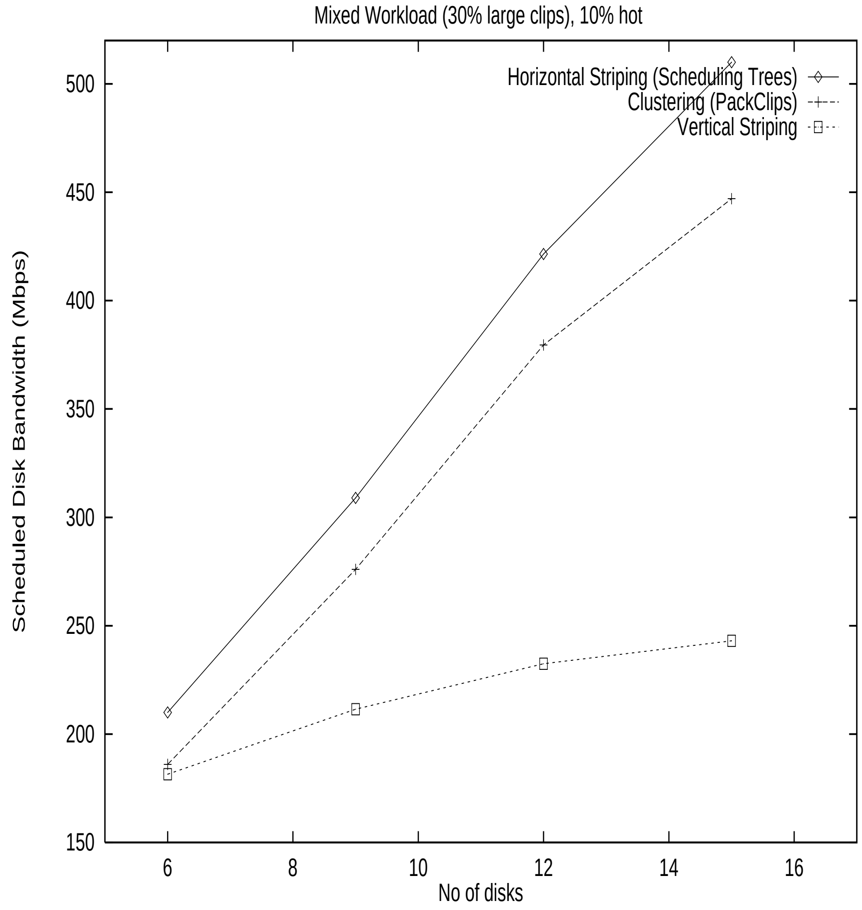


Experimental Results (cont.)

- Short Clips, 50% "Hot"



- Mixed Clips (30/70), 10% "Hot"



Conclusions and Future Work

- HS + Periodic Scheduling is better
- General framework and methodology for a large class of periodic scheduling problems
- Extensions: periodic scheduling can benefit other layout and storage allocation strategies

What about Memory?

- Additional buffers allow more flexibility for stream data retrieval
- Connections to the *Pinwheel Problem* and *Distance-Constrained Task Scheduling*
- Shared memory: Buffer management issues for competing streams

EPPV under Clustering (more...)

- Heuristic Solution: PackClips

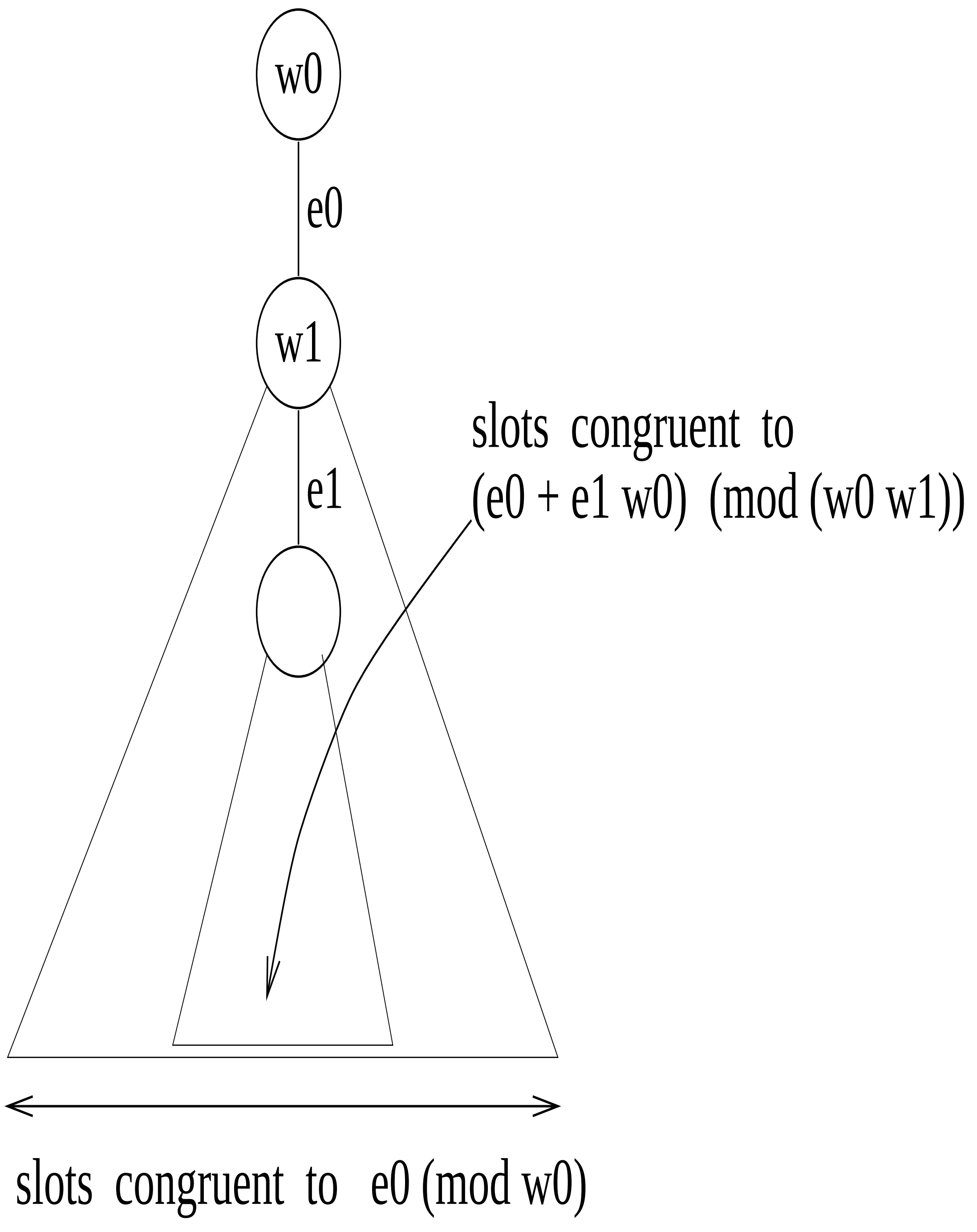
1. Sort C in decreasing order of *value density* = value()/size()
2. Pack into bins using First-Fit
3. Select the n_{disk} most "valuable" bins in the final packing

Bandwidth & Storage Constraints?

- 2-dimensional size() vector = [BANDWIDTH requirement, STORAGE requirement]
- *value density* = value()/max.component{size()} and 2-d vector packing

Theorem: (1) 1-d PackClips is 1/2-approximate (2) 2-d PackClips is 1/4-approximate, assuming STORAGE requirement $\leq 1/2$

Scheduling Tree: some intuition...



Experiment Parameter Settings

Disk Params.	
r_{disk}	80 Mbps
c_{disk}	4 GBytes
t_{seek}	24 ms
t_{lat}	9.3 ms
T	1 sec

Workload #1	
l_i	90 – 120 min
T_i	40-60 , 150-180 min
r_i	1.5 Mbps
No. clips	20 – 200
Hot clips	5% – 50%

Workload #2	
l_i	2 – 10 min
T_i	20-30, 40-60 min
r_i	2 – 4 Mbps
No. clips	80 – 200
Hot clips	5% – 50%