Storage Systems Management

Guillermo Alvarez, Kim Keeton, Arif Merchant, Erik Riedel, and John Wilkes

Hewlett-Packard Labs,
Storage Systems Program

Tutorial overview

♦ Introduction
  – Why storage is important
  – Customer problems
  – Case study – DSS database server
  – The storage management market
♦ Storage Systems 101 – the building blocks
♦ Major problems in storage management
♦ Current solutions
♦ Our vision
♦ Research challenges
♦ Conclusions
**Introduction – why do we care?**

- **Storage systems**
  - the place where persistent data is kept
  - the center of the universe!

- **Why?**
  - information (and hence storage) is key to most endeavors
  - storage is big business (tens of $billions per year)
  - sheer quantities (hundreds of petabytes per year)

  - "Storage will dominate our business in a few years"
    - Compaq VP, 1998
  - "In 3 to 5 years, we will start seeing servers as peripherals to storage"
    - SUN Chief Technology Officer, 1998
  - "We’ll plug into whatever servers you have"
    - IBM Versatile Storage Server ad, 1999

---

**Introduction – storage hierarchy**

- **Primary storage: CPU**
  - registers (1 cycle, a few ns)
  - cache (10-200+ cycles, 0.02–0.5us)
  - local main memory (0.2–4us)
  - NUMA memory (2–10x local memory)

- **Secondary storage (online storage)**
  - magnetic disks (2–20ms)
  - solid state disks (0.05–0.5ms)
  - cache in storage controllers (0.05–0.5ms)

- **Tertiary storage**
  - removable media: tape cartridges, floppies, CD, ... (ms to minutes)
  - tape libraries, optical jukeboxes (nearline) (few s to few minutes)
  - tape vaults (few minutes to days)
Customer problems – complexity

- Need more capacity.
- Need better performance.
- Need high availability.
- Must rebalance the load.
- Must add devices.

UGH!... my head hurts!

- Quality of service guarantees.
- Network attached storage.
- More demanding applications.

AAAGH!... Brain exploding!

Headache today? Migraine tomorrow!!

Customer problems – scale

- System scale is exploding
  - Information density is dropping
    - text files >> DBMS >> data mining >> images >> email >> multimedia ...
  - Sheer numbers of applications, host systems, devices
  - Rate of growth
    - sometimes wildly unpredictable
- Growing demands from business side
  - continuous availability
  - predictable, stable performance
  - lower costs
- Not enough skilled people
**Customer problems – knobs**

Too many knobs!

- RAID 3 data layout, across 5 of the disks on array F, using a 64KB stripe size, 3MB dedicated buffer cache with 256KB sequential read-ahead buffer, delayed write-back with 1MB NVRAM buffer and max 10s residency time, dual 256KB/s links via host interfaces 12/4.3.0 and 16/0.4.3, 1Gb/s trunk links between FibreChannel switches A-3 and B-4.
- Business-critical availability
- 150 i/o per sec
- 200ms response time

**Customer problems – cost**

- Storage is a big piece of the pie
  - 30-50% of total system cost in storage
  - And that’s before you pay for management!

<table>
<thead>
<tr>
<th>Normalized benchmark costs ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1997 TPC-D</strong> (HP-UX, 300GB scale factor)</td>
</tr>
<tr>
<td>37% storage</td>
</tr>
<tr>
<td>42% CPUs</td>
</tr>
<tr>
<td>21% software</td>
</tr>
</tbody>
</table>
Case study – DSS database server

- Hewlett-Packard N-class TPC-H Server
  - HP 9000 N4000 Enterprise Server
  - Informix Extended Parallel Server database
  - 8 x 550 MHz PA-RISC processors
  - 32 GB memory
  - 3 SureStore E Disk Array FC60s
    - 28 x 18.2 GB disks each in RAID1 (mirrored)
    - tables & indices
  - 4 SureStore E Disk System SC10s
    - 9 x 18.2 GB disks each in RAID0 (JBOD)
    - temporary space
  - 2.1 TB total storage (111 disks)
  - $1,154,133 total cost, $457,984 storage cost
  - 1,592 QphH@300GB, $973 / QphH@300GB

Storage management market (DataQuest)

- Storage infrastructure
  - basic data organization
  - file systems, volume mgmt, physical replication
  - who: various OS file systems (everyone does it), Veritas

- Data management
  - backup, restore, archive, HSM
  - who: Legato, IBM ADSM/Tivoli™, HP, CA Unicenter™, EMC, Sun

- Enterprise storage management
  - everything else
  - “management of various storage resources on the network including [disk, tape]…”
  - who: IBM/Tivoli™, HP SureStore™, Compaq SANworks™, CA Unicenter™, HighGround, BMC, CommVault
Market trends – storage management

- **Total market**
  - $2.6 billion in 1998, $6.6 billion by 2003

- **Three areas**
  - storage infrastructure
  - data management
  - enterprise storage resource management

### Outline

- **Introduction**
- **Storage Systems 101 – the building blocks**
  - Disk drives
  - Disk arrays
  - Storage area networks (SANs)
  - Network-attached storage (NAS)
- **Major problems in storage management**
- **Current solutions**
- **Our vision**
- **Research challenges**
- **Conclusions**
Disk drive – what’s inside?

- Spindle
- Arm
- Actuator
- Platters
- Electronics
- SCSI

Image courtesy of Seagate Technology, Inc.
Original material Copyright © 2000 Seagate Technology, Inc.

Disk drive – platters

- 1 to 12 platters per disk
- 2 heads per platter
- 2,000 to 40,000 tracks per platter
- 50 to 200 kilobytes per track
- 512 bytes per sector

Two sides, write on top and bottom

Areal density = linear density * track density
**Trends**

*Track, Areal, Linear Density Perspective*

- Track Density, CGR = 30%
- Areal Density, CGR = 65%
- Linear Density, CGR = 23%

**IBM Disk Drive Products**
- Circles = Server products
- Squares = Mobile products

*Ed Grochowski at Almaden*

---

**Disk drive – electronics**

*Quantum Viking (circa 1997)*

- 6 Chips
- R/W Channel
- uProcessor
  - 32-bit, 25 MHz
- Power Array
- 2 MB DRAM
- Control ASIC
  - SCSI, servo, ECC
- Motor/Spindle
  - Connect to disk
  - Control processor
  - Cache memory
  - Control ASIC
  - Connect to motor

*2000-06-SigmetricsTutorial, 15*
*Storage Systems Program*
Disk drive – on-disk controller

- Caching
  - read-ahead
  - write behind (careful!)
  - atomicity guarantees (not!)

- Controlling the mechanism
  - head scheduling
  - spindle motor
  - servo tracking

- Data path management
  - protocol sequencing
  - request scheduling

Why disk arrays?

Because disks are slow.

AND

Because stuff happens.
Problem – failures happen

- Things break -- in a moderately predictable way in aggregate

![Relative mortality rate graph](image)

- Metrics:
  - MTTF: mean time to failure -- a rate, not a period
  - AFR: annual failure rate (better – but still just middle of “bathtub”)
  - MTTR: mean time to repair

Solution – redundancy

- Complete copies
  - replication, mirroring

- Partial redundancy
  - Hamming codes/ECC
    - tolerates mangling of elements
    - unnecessarily strong – we know when disks are broken

- Parity
  - XOR sets (stripes) of data blocks to calculate a parity block
  - any data block can be reconstructed from the others + parity

Mirror copies

Parity unit (xor of rest of stripe units in same stripe)
How redundancy helps

- Individual disk drives
  - originally (mid-1980s), these were among the most unreliable components in a system
  - nowadays, they are one of the more reliable ones (AFR of 1 to 2%)
  - but failure rates are proportional to numbers ...

- Assumes independent failures
  - warning! danger! caution! error!

- With no redundancy ...
  \[ AFR_{disks} \sim N_{disks} \times AFR_{disk} \]

- With one degree of redundancy ...
  \[ AFR_{raid} \sim AFR_{disks}^{N_{disks}} \times MTTR_{disk} \times AFR_{disks}^{N_{disks}-1} \]

Downsides of redundancy

- Cost
  - replicating everything costs 2x as much storage
  - solution – partial redundancy

- Slower updates
  - 2x as many copies to write to
  - ... even worse with partial redundancy

- Greater complexity
  - 80 - 90% of disk array firmware is error handling
  - lots and lots of configuration choices ...
Disk array taxonomy

RAID = Redundant Arrays of Inexpensive Disks

Currently accepted RAID levels:
- 0: no redundancy (JBOD)
- 1: full copy (mirroring)
- 10: striped mirrors
- 2: Hamming-code/ECC (not used)
- 3: byte-interleaved parity
- 4: block-interleaved parity (more useful variant of RAID3)
- 5: rotated block-interleaved parity
- 6: double parity (“P+Q parity” -- rare)

RAID levels 0, 1, 10

- **RAID0**: striping (no redundancy)
  - *Striping* balances the load and allows large transfers to happen in parallel

- **RAID1**: aka mirroring (full redundancy)
  - *Mirroring* gives 2x the read bandwidth per disk, but writes have to go to both

- **RAID10**: striped mirroring (full redundancy)
**RAID level 5**

- **RAID5** - rotated-parity striping to balance the load

- **Updating parity is expensive for small writes**
  - write-caching becomes especially important

  ![XOR diagram](image)

  1. Read old data & parity
  2. Compute new parity
  3. Write new data + parity

  => 4x I/O operations per small write

  Rotating the parity balances the parity load across all the disks; striping allows fast large transfers.

RAID5 is the configuration of choice for all but performance-intensive workloads.

---

**Disk array – mid-range architecture**

- **Mid-range array** (e.g., HP FC60)
  - sometimes separate controller and disk boxes
  - up to 1 to 2 TB disk, 0.5 GB cache memory
  - can saturate a 100 MB/s FibreChannel link; O(10,000 IOs/s)

![Disk array diagram](image)

Packaging:
- whole array is in a single box, or
- array controller is in separate box
**Disk array – high-end architecture**

<table>
<thead>
<tr>
<th>Packaging:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• array controller in a separate box (disks in rack-mountable trays), to</td>
</tr>
<tr>
<td>• array controller is one of several 6' racks</td>
</tr>
</tbody>
</table>

- up to a few TB disk, a few GB cache |
- can saturate a few 100 MB/s FibreChannel links |
- O(50,000+ IOs/s) |
- lots of caching |
- multiple host interfaces |
- quality power, cabling, cooling |
- phone-home, remote mgmt, support infrastructure |

---

**Disk arrays – logical units**

- **Arrays provide multiple LUNs (SCSI Logical UNIts)**
  - basic unit of control, management |
  - one or more disk drives bound together into a common layout (choose a RAID level) |
  - different LUNs can have different sizes, different layouts |
  - 8 to 32 LUNs at the mid-range |
  - thousands of LUNs at the high-end |
    - SCSI limit: 4096 LUNs, from a 12 bit LUN identifier |

- **A few common variations (there are many more)**
  - parts of disks instead of whole disks |
  - LUNs may be named relative to ports, not uniquely |
  - LUNs can have different caching behavior
Why storage networks?

Because we want to *share* storage?*

*Sharing isn’t always easy.

Storage Area Networks – FibreChannel

- Arbitrated Loop (FC-AL)
  - shared, dual loop

- Switched Fabrics
  - dedicated ports
  - hosts, disks, arrays
  - high aggregate bandwidth

- Same Physical Layer
  - 100 MB/s bandwidth
  - copper: coax, or backplanes
  - fibre: up to 500m multimode; 10km single-mode

- SCSI, IP encapsulations
  - FC Control Protocol (FCP)
  - SCSI over IP
LAN vs SAN vs NAS?

- **Network hardware:** FibreChannel vs Ethernet
  - 1 Gb/s E’net available today, 2 Gb/s FC ready
  - 10 Gb/s E’net will (probably) be ready first

- **Storage interface:** blocks vs “files”
  - block storage devices (SCSI)
  - object-based storage (under discussion)
  - **NAS** => file servers (Netware, NFS, CIFS)

- **Network protocol:** FCP vs TCP/IP
  - specialized protocol vs. general-purpose one

- **SAN (Storage Area Network)**
  - dedicated network, used (largely) for storage
  - **whatever the protocol!**

Open issue – blocks vs. files?

- **Blocks (SCSI)**
  - critical path simple => fast
  - very “simple” interface
  - hard to push function to storage device

- **Files (Netware, NFS, CIFS)**
  - can optimize layout and caching
  - finer-grained protection possible
  - critical path longer => slower

---

Disks

- **Locally attached storage**
- **Distributed file/dbms system**
- **Shared storage pool plus distributed FS**
Outline

- Introduction
- Storage Systems 101 – the building blocks
- Major problems in storage management
  - System design and configuration (device management)
  - Problem detection and diagnosis (error management)
  - Capacity planning (space management)
  - Performance tuning (performance management)
  - High availability (availability management)
  - Automation (self-managing storage)
- Current solutions
- Our vision
- Research challenges
- Conclusions

System design and device configuration

- How to decide which storage devices to buy
  - how many?
  - what kind?
    - how fast, how big
  - how are they connected?
    - SCSI, FC-AL, switches, SAN, NAS
- How to set device configuration parameters
  - RAID level?
    - RAID0, RAID1, RAID10, RAID5
  - disks per stripe?
  - stripe size?
  - buffer management?
  - prefetch and writeback policies?
    - aggressive, conservative
Problem detection and diagnosis

- What must be monitored to detect device failures?
  - across hosts, arrays, networks
  - across multiple vendors
  - across multiple operating systems
- What system information must be available to diagnose root cause?
  - isolate problems
- What capabilities must be available to correct problems?
  - redundancy (RAID levels)
  - multiple network paths
  - transparent failover
  - replacement parts (hot spares)

Capacity planning

- How to keep up with users’ capacity demands?
  - tracking growth
  - predicting growth
  - acquiring additional storage
  - installing and configuring additional storage
  - identifying hot vs. cold data
  - often tied closely to performance
  - variance in usage patterns
Performance tuning

- How should the storage system be designed to maximize performance?
  - LUN design
  - logical volume design
  - file/database layout onto logical volumes
- What must be monitored to detect performance bottlenecks?
- How do we translate between different levels of abstraction?
  - LUNs vs. logical volumes vs. database table
  - blocks vs. files
- Service level agreements (SLA and QoS)
  - specify customer business requirements
  - “enforce” service levels

High availability

- What design must we use to avoid single points of failure?
- What RAID levels must be used to achieve desired availability?
- Reliability
  - \( R(t) \) = likelihood system up continuously from time 0 to time \( t \)
- Availability
  - \( A(t) \) = likelihood system will be up at time \( t \)
- Performability
  - \( P(t,p) \) = likelihood system will be providing performance \( p \) at time \( t \)
Automation

- How do we make all this happen with minimal human involvement?
  - remove the human from the loop whenever possible
- High-level goals
  - what to do, not how to do it
  - set and forget
- Manipulate device knobs
- Automatic performance analysis
- Service level agreements
- Grow/shrink as necessary
  - capacity and performance
- Transparently

Outline

- Introduction
- Storage Systems 101: the building blocks
- Major problems in storage management
- Current solutions
  - Storage management products
- Our vision
- Research challenges
- Conclusions
Storage management products

- **Pre-sales tools**
  - system design and device configuration
  - capacity planning
  - high availability

- **IBM Tivoli™**
- **Compaq SANworks™**
- **HP SureStore™ SAN Manager**
- **CA Unicenter™**
  - problem detection and diagnosis
  - high availability
- **HighGround Storage Resource Manager™**
- **BMC Patrol™**
  - problem detection and diagnosis
  - performance monitoring

Outline

- **Introduction**
- **Storage Systems 101 – the building blocks**
- **Major problems in storage management**
- **Current solutions**
- **Our vision**
  - Stress-free storage
  - The storage utility
- **Research challenges**
- **Conclusions**
Stress-free storage

A storage system for the enterprise made from:

- High-quality individual components
- *Management software* to glue it all together
- Result – easy-to-use, easy-to-manage system that delivers business goals
  - reduce personnel to a single person and a dog
  - have *complain* and *commend* buttons for each user

The storage utility

- Attribute management: *what to do, not how to do it*
- Distributed storage manager: dynamically-mapped, scalable, host-independent storage, online data migration
- Network attached storage devices: QoS, security, smart devices
The storage utility – how is it done?

Just a few Big Ideas ...

- **Goal-directed self-management**
  - specify what to do (goals), not how to do it (implementation)

- **Automatic (re)design and (re)configuration**
  - to reduce complexity & human effort

- **Predictable behavior through guarantees**
  - QoS = performance + availability + cost

- **Software as the key differentiator**

---

Customer lifecycle

(Changing) business requirements

- Monitor
- Design / redesign
- Configure / reconfigure
- Storage utility
Research challenges from lifecycle

- How to characterize application I/O behavior?
- Workload modeling

- Storage device modeling
- How to predict device performance for a workload?

- System (re)design & (re)configuration
- How to automatically design a system and assign data to devices?

- Monitoring
- How to monitor performance & availability?

- Runtime system
- How to support online migration and QoS enforcement?

Outline

- Introduction
- Storage Systems 101 – the building blocks
- Major problems in storage management
- Current solutions
- Our vision
- Research challenges
  - Workload modeling
  - Storage device modeling
  - Initial system design
  - System configuration
  - Online system monitoring
  - System reconfiguration
  - Some additional challenges
- Conclusions
Customer lifecycle – the (re)design step

(Changing) business requirements → Workload modeling → Configure / reconfigure → Storage utility → Monitor → Design / redesign → Configure / reconfigure → Workload modeling

Research challenge – workload modeling

**Goal:** succinctly capture important workload characteristics

- **Requirements**
  - capacity, availability
  - response time, b/w, I/O rate
- **Behaviors** (use patterns)
  - spatial locality (hot spots, reuse)
  - temporal locality (burstiness)
  - request size

- **QoS, Service Level Agreements, or guarantees**
  - If application behaves *this way*, storage system will provide *this much* service.
Workload modeling - SSP approach

- **Declarative** specification of workload attributes
- **Workload** = set of **stores** + **streams**
  - Stores capture static requirements (e.g., capacity)
  - Streams capture dynamic workload requirements (e.g., bandwidth, availability) to a store
- *(Simplified) workload unit example:*
  Store store0 { capacity 1e9 (bytes) }
  Stream stream0 {
    boundTo store0
    requestRate {ARW 800 600 200} (request/sec) requirements
    requestSize {ARW 4096 4096 4096} (bytes)
    sequentialRunCount {mean-variance 20 5} (requests) use patterns
    onTime 90 (seconds) offTime 99 (seconds)
    overlapFraction {stream1 1.0} {stream2 0.0}
  }

Workload modeling – Rubicon

- **Workload characterization**
  - evaluate requirements and behaviors of applications
- **Device characterization**
  - measure device performance
- **System monitoring and tuning**
  - spot storage bottlenecks and evaluate design changes
- **System validation (together with Pylon)**
  - compare effects of synthetic (replayed) workload vs. original measurements
Workload modeling – case study

- Decision support (DSS) database
  - Oracle
  - 300 GB TPC-D database
  - Presentation focus: TPC-D Q5

Also examining:
- File system
- Email server
- Online transaction processing (OLTP) database
- Enterprise resource planning (ERP) database
- Scientific applications
- Web server
Workload modeling – request size

- Dominated by larger (64KB) reads to tables
- Different behavior from different parts of the database:
  - table, indices, temp space, log

Workload modeling – I/O phasing

- Request rates vary widely
- Most multi-table queries have multiple phases
  - different tables active during different periods
Workload modeling – read:write ratio

♦ “Read-only” workload exhibits writes!
  – read:write ratio varies over phases

![Graph showing read percentage over elapsed time](image)

Workload modeling – lessons learned

♦ Lessons learned:
  – list of important characteristics is longer than you think
  – distributions, not averages, are important

♦ Some characteristics of interest:
  – request size distribution
  – request rate distribution
  – read:write ratio
  – spatial locality (e.g., sequentiality)
  – temporal locality (e.g., data re-references)
  – phased behavior
  – correlation between accesses to different parts of storage system
  – burstiness
Workload modeling – related work

Workload characterization case studies
- File system tracing
  - [Ousterhout85, Miller91, Ramakrishnan92, Baker91, Gribble98]
- Network tracing
  - [Caceres91, Paxson94, Paxson97]
- I/O tracing
  - [Bates91, Ruemmler93, Gomez98, Hsu99]

Tools
- Offline trace gathering, analysis and visualization
  - [Grimsrud95, IBM99]
- Extensible trace analysis
  - Tramp [Touati91]
- Network packet filters
  - [Mogul87, McCanne93]
- Trace visualization
  - [Heath91, Malony91, Hibbard94, Eick96, Aiken96, Livny97]

Issues in workload modeling

- What characteristics should we measure?
  - for workload regeneration
  - for QoS specification
  - for device performance prediction
- How to quantify these characteristics?
  - what metrics, and in how much detail?
  - ex: correlations, burstiness, spatial and temporal locality
- What’s the relative importance of these properties?
- How to model the scaling behavior of applications?
  - ex: number of users, size of database
- How to provide semantic mapping between application operations and storage system requirements?
**Issues in workload modeling (cont.)**

- How much does behavior vary between different apps running same workload?
  - ex: Oracle vs. Informix vs. DB2 vs. SQLServer
  - ex: NFS vs. CIFS
- How to model distributed applications and their interactions?
- How does NAS file workload characterization differ from block-oriented I/O characterization?

---

**Customer lifecycle – the (re)design step**

- Workload modeling
- Device modeling
- System design

(Changing) business requirements → Design / redesign → Configure / reconfigure → Monitor → Device modeling → Storage utility

---

Storage Systems Management
Research challenge – device modeling

Goal: capture storage device characteristics in a predictive model:
- Capabilities
  - performance: transfer rate, positioning time, caching, ...
  - capacity
  - failure model
- Configuration options
- Costs

Device modeling – SSP approach

- Fast, analytic models of device behavior
- Storage system = set of hosts + devices + fabric(s)
  - Hosts: where work is generated
    - (probably) support logical volume manager
  - Storage devices
    - provide LUNs (onto which workload stores/shards get mapped)
    - have capabilities (performance, capacity, availability) + cost
  - Storage fabric: connects hosts to storage devices
- (Simplified) device model example:
  - available device capacity > \( \Sigma \) capacity_store_i
  - available bandwidth > \( \Sigma \) requestRate_stream_i * requestSize_stream_i
  - for streams that are “on” together
Device modeling – building the model

- How to deal with:
  - non-deterministic experiments
    - e.g. measuring cache IO/s bandwidth → known error bars
  - desired results that aren’t the outcome of any single experiment
    - solve the system of equations to get results
  - reconfiguring a device between different experiments can be time-consuming
    - very long process → failures must be tolerated (restart)
    - next point to consider may depend on outcomes of previous experiments
    - how good are a model’s predictions?

- SSP approach: Pacioli
  - measurement of device-specific performance characteristics
  - validating complete models against the real system

Pacioli structure

Desired confidence
Experiments
Factor domains
Measurement
Equations (dependencies)
Reconfiguration costs
Model parameters
Black-box model
Goodness of fit (pass/fail)
Validation

Future work: automatic calibration
Device modeling – related work

- Ruemmler and Wilkes, 1993
  - accurate disk drive simulation model – prioritized components
  - detailed characteristics for two disk drives
- Worthington, et al., 1995
  - Black-box techniques for empirically extracting SCSI disk parameters
- Shriver, et al., 1997
  - disk drive model creatable by composing models of individual components
  - performance prediction dependent on input workload and predictions of lower-level models
- Pythia [Pentakalos, et al., 1997]
  - automatically builds and solves analytic model of storage system
  - inputs: graphical representation of system and workload
  - Pythia/WK: uses clustering algorithms to characterize workloads
- Disk arrays
  - [Thomasian94, Merchant96, Menon97]

Issues in device modeling

- What properties does the model need to capture?
  - to utilize workload characteristics
  - for accurate vs. fast predictions
- What's the relative importance of these properties?
- What's the right tradeoff between model accuracy and performance?
  - for simulations
  - for input to optimization
  - set of increasing fidelity device models
- Do we need to model hosts/servers to model storage system behavior adequately?
- (How) can we automatically extract model parameters?
- How to create device models that can use very complex workload characteristics?
  - ex: fractal characteristics
- How to incorporate availability/performability into models?
- How to model NAS devices?
Customer lifecycle – the (re)design step

(Changing) business requirements

Workload modeling
Device modeling
System design

Design / redesign
Configure / reconfigure

System (re)design

Monitor

Storage utility

System design & assignment problem

Applications
workload

workload requirements

Storage utility

Assignment engine (solver)

storage-system configuration

storage device abilities
Problem:
- convert workloads, business needs and device characteristics into assignment of stores and streams to devices

SSP approach: Forum
- constraint-based multi-dim. bin-packing

Sample constraints:
- number of devices store assigned to = 1
- sum of store sizes <= capacity
- sum of stream utilizations <= 1.0

Sample objective functions:
- minimize cost
- balance load

Forum basics

- Concise workload models
  - sources:
    - library of models for common workload types
    - automatically characterized from running workload (Rubicon)

- Fast, acceptable-fidelity device models
  - executed in inner loop of optimizer
  - source: library of storage-device characterizations

- Search-space exploration algorithms
  - heuristics for trying “what if?”
    - good news: simple ones work well
  - utility-based objectives, modulated by business goals
    - minimum cost, maximum availability, balanced load, greater growth space, …
Initial system design – disk arrays

Problem:
- extending the single disk solution (Forum) to disk arrays
- the space of array designs is potentially huge:
  - LUN sizes and RAID levels, stripe unit sizes, disks in LUNs, prefetch multiplier and water marks, cache page size, read/write cache, ...
  - more work needed before the Forum solver can run

SSP approach: Minerva

Basic Minerva modules:
- **Tagger**: tag stores with their type (RAID1, RAID5)
- **Allocator**: estimate how many arrays needed to support this
- **Design procedure**: configure each of the allocated arrays
- **Forum solver**: map stores to LUNs
  [repeat until complete]
- **Cleaner**: prune any unnecessary resources
- **Optimizer (Forum solver)**: can rearrangements decrease the cost or better balance the load?

Minerva – how the modules work

- **Tagger**: rules tag each store according to how it’s accessed
  - if capacity-bound, RAID5
  - if read-mostly, RAID1/0
  - ...
- **Allocator and designer**: based on aggregate workload, buy and configure arrays that can do the job
  - find cheapest set that a priori may work
- **Forum solver**: assign stores to LUNs
- **Cleaner**: discard disks, cabinets, busses, ... that service only empty LUNs
- **Optimizer**: use Forum solver with different objective functions to generate alternative solutions; then pick best
  - mincost on final set: can cost be reduced further?
  - optimize load balancing (utilization)
Minerva – disk array system design

A decision-support and order-entry workload

Select RAID store type

Select and configure array(s)

Assign data to arrays

Optimize array layout

Retry with any unassigned stores

Optimized array design

Configured array: data laid out on the array’s LUNs

Array template: a rule set for the different arrays that can be built

Workload tagged as RAID1 (solid) and RAID5 (striped)

Initial system design - related work

✦ Storage management [Gelb89]
  – Logical view of data separate from physical device characteristics – simplifies management

✦ File assignment
  – Files placed on storage devices with aim of optimizing objective(s)
  – [Dowdy82, Wolf89, Pattipati90, Awerbuch93]

✦ Optimization algorithms
  – Bin-packing heuristics [Coffman84]
  – Toyoda gradient [Toyoda75]
  – Simulated annealing [Drexl88]
  – Relaxation approaches [Pattipati90, Trick92]
  – Genetic algorithms [Chu97]
**Issues in system design and allocation**

- What optimization algorithms are most effective?
- What optimization objectives and constraints produce reasonable designs?
  - ex: cost of reconfiguring system
- What's the right part of the storage design space to explore?
  - ex: RAID level vs. stripe unit size vs. cache mgmt parameters
- What are reasonable general guidelines for tagging a store's RAID level?
- What (other) decompositions of the design and allocation problem are reasonable?
- How to generalize system design?
  - for SAN environment
  - for host and applications

---

**Customer lifecycle – build/configure step**

- (Changing) business requirements
- Design / redesign
- Configure / reconfigure
- Building the system described in design step
- Monitor
- Storage utility

Building the system described in design step
Research challenge – configuration

SSP approach: Panopticon
- converts OS-neutral designs into OS-specific scripts and performs configuration commands

Issues in configuration

- How to do system discovery?
  - ex: existing state, presence of new devices
  - dealing with inconsistent information
  - in a scalable fashion

- How to abstractly describe storage devices?
  - for system discovery output
  - for input to tools that perform changes

- How to automate the physical design process?
  - ex: physical space allocation, wiring, power, cooling
Putting it all together – TPC-D case study

**Workload measurements**

**Workload specification**

**Design**
(OS-neutral description of what to put on each storage device, and how to configure them)

**Configuration**
(LVM scripts, disk array LUNs, ...)

**Validate**
(see next slide)

---

**Putting it all together – TPC-D case study**

- Initial design: human expert
  - 30 disks
  - measured workload used as Minerva input
- Minerva design: automatic!
  - 16 disks
  - same performance (within 3%)

---

![Chart showing query execution times](chart.png)
Customer lifecycle: system monitoring

(Changing) business requirements

Design / redesign

Configure / reconfigure

Monitor

Research challenge – online monitoring

SSP approach: Rubicon

Related work: online performance monitoring
  – Paradyn [Miller95] and Pablo [Reed93]
Issues in online monitoring

- What quantities must be monitored?
  - to detect component failures
  - to detect performance bottlenecks
  - to enforce QoS requirements/detect QoS violations
  - to detect performance trends
- How to monitor in a scalable fashion?
- How to monitor in a flexible fashion?
  - ex: attributes that are specific to one type of device
- How to translate between different levels of abstraction?
  - ex: LUNs vs. logical volumes vs. database tables
- What policies and thresholds should be used for generating alarms?

Customer lifecycle – cont. improvement

(Changing) business requirements

Design / redesign

Configure / reconfigure

Monitor

Storage utility
Research challenge – reconfiguration

♦ System should automatically respond to workload and device needs to meet user performance and availability goals

Reconfiguration Triggers:
- Alarm reports, device failures
- Offered load analysis/trend
- QoS analysis/trends
- Business-need changes
- Technology changes

Change threshold: go, no-go?

Redesign process

Reconfiguration process

Storage utility

Equipment tool
- LVM setup tool
- Service Level Agreement
- Array configuration tool
- LUN security tool
- System discovery tool
- Fabric configuration tool
- Migration planner tool
- Migration conductor

Migration conductor

2000-06-SigmetricsTutorial.86
Storage Systems Program

2000-06-SigmetricsTutorial.87
Storage Systems Program
Research challenge – reconfiguration

- Events trigger redesign decision
  - How do we decide when to reconfigure?
- Solver creates new system assignment
- Reconfiguration inputs:
  - current system configuration/assignment
  - desired system configuration/assignment
- 1. Build a migration plan
  - How to devise a plan for data movement with general constraints?
    - ex: capacity, performance, availability
  - How to generalize for variable-sized data?
  - How to allow parallel execution?
  - How much free space is needed?
- 2. Make it happen – online
  - Runtime system: combination of host-side virtualization, metadata management, and storage device hooks

Customer lifecycle – the running system

(Changing) business requirements

Design / redesign

Configure / reconfigure

Monitor

Virtualization
Robust metadata
Security enforcement
QoS enhancement

Storage utility
Palladio – SSP’s runtime system approach

- Automatic responses to system load changes
  - goal-directed, not policy-based
  - mechanisms for attribute management
- Key issues
  - How to provide online data migration?
    - “virtualization” of metadata
    - mechanisms for online data migration, replication
  - How to provide self-management?
    - automatic inclusion of new resources
    - automatic failure handling
  - How to recover from disasters?
    - robust metadata management
    - multiple site support
  - How to enforce security and QoS in shared environment?

Palladio system architecture
Research challenge – runtime system

- Ensuring metadata is always available
  - Even in the face of network partitioning [Golding99]
- Managing concurrency at the large scale
  - Optimistic concurrency control protocols [Amiri00]
- Enforcing security in a multi-host environment
  - Has to be done directly at storage device in a shared-resource environment
  - Carnegie Mellon NASD [Gobioff99, Gibson98]
- QoS enforcement (e.g. Service Level Agreements)
  - How should these be specified?
  - What portions should be enforced by which component?
  - How can violations be detected? Handled? At what cost?
  - [Golubchik99, Bruno99, Wijayaratne00]

Runtime system related work

- CMU network-attached disks
  - disks present file-like objects
  - many disks aggregated to make system
  - [Gibson97, Gibson98]
- Distributed storage service
  - MIT Logical disks [deJonge93]
  - Compaq/DEC SRC Petal [Lee96]
  - U of Arizona Swarm [Hartman99]
- Distributed file systems
  - CMU Andrew FS [Howard88]
  - Berkeley Zebra [Hartman93]
  - Berkeley xFS [Anderson95]
  - Compaq SRC Frangipani (FS for Petal) [Thekkath97]
Additional research challenges

- How do we design SAN fabrics automatically?
- What’s the right interface for storage?
  - files vs. blocks
  - NAS vs. SAN
  - how do we ensure secure storage?
  - how much does this matter for storage management?
- How can we exploit device intelligence to make storage management easier?
- How do we describe maintainability and availability?

SAN fabric design

- Problem description
  - given: flows betw. endpoints and SAN characteristics
  - return: set of internal nodes and node-node links (incl. flows)
  - must satisfy:
    - flow requirements, link and node constraints, connectivity constraints
- Current state of the art
  - designs are done by hand, using a few simple topologies
- Automation hasn’t proven straightforward
  - degree-constraints seems unusual
  - divide-and-conquer seems unhelpful
- “Extra credit” items are very important
  - fault tolerance: designing for all possible failure cases
  - multiple layers of switches/hubs possible
**Storage interface – blocks vs. files?**

- **Blocks (SCSI)**
  - critical path simple => fast
  - very “simple” interface
  - hard to push function to storage device

- **Files (Netware, NFS, CIFS)**
  - can optimize layout and caching
  - finer-grained protection possible
  - critical path longer => slower

---

**Exploiting device intelligence**

- **Observations**
  - processing capabilities, memory capacity, and networking ability of storage devices increasing
  - aggregate computational ability and aggregate bandwidth at devices are greater than at central processors

- **Goal**
  - use storage devices to run application code and improve performance of data-intensive applications

- **Focus to date**
  - file system functionality in devices
    - [Wilkes92, Cao93, Wang99]
  - database and data processing functionality in devices
    - [Keeton98, Riedel98, Acharya98, Uysal00, Riedel00]
    - revisits database machine work from late 1970s – early 1980s

- **Potential future work**
  - storage management functionality in devices
    - ex: data migration, resource discovery and mgmt, monitoring
Describing manageability & availability

❄ Observations
  – computer architecture and operating systems community shift in research interest: non-performance topics
  – difficulty of maintaining large systems

❄ Goals
  – enumerate important factors in managing large systems
  – describe (quantitative) metrics for evaluating system manageability/maintainability

❄ Initial efforts
  – availability metrics [Brown00]

Summary – storage mgmt challenges

❄ Workload characterization/modeling
❄ Storage device modeling
❄ Initial system design
❄ System configuration
❄ Online system monitoring
❄ System reconfiguration
❄ Runtime system
❄ SAN fabric design
❄ Storage system interfaces
❄ Exploiting smart devices
❄ Describing/Quantifying manageability
Summary – underlying trends

- Commoditization of hardware
  - software+services are the real differentiators, not hardware

- Network upheavals (FC, Infiniband, 1-10Gb’s Ethernet, IP)
  - Internet protocols becoming dominant (“when, not if”)
  - block servers -> file abstractions (whether/when?)

- Cheap distributed CPU cycles
  - storage appliances, smart storage devices, function shipping

- Demands for predictability (aka QoS)
  - guarantees for availability, performance, security

- The services revolution
  - rent-a-Terabyte?

Conclusions – key ideas

Just a few Big Ideas:
- Goal-directed self-management
- Automatic (re)design and (re)configuration
- Predictable behavior through guarantees
- Software as the key differentiator

(Changing) business requirements

Design / redesign
Configure / reconfigure
Monitor

Storage utility

Storage Systems Management
Conclusions - the storage utility

Attribute management
• what to do, not how to do it

Distributed storage manager
• dynamically-mapped, scalable, host-independent storage
• online data migration

Network-attached storage devices
• QoS, security, smart devices

Conclusions

♦ Storage systems represent an interesting technical (and commercial) opportunity
  – data is important to people
  – large scale
  – very high performance
  – extreme availability/fault-tolerance needs

♦ Rich storage-related research topics
  – optimization problems
  – measurement and modeling problems
  – distributed systems problems
Acknowledgements

- **SSP:** Eric Anderson, Ralph Becker-Szendy, Michael Hobbs, Cristina Solorzano, Susan Spence, Ram Swaminathan, Simon Towers, Mustafa Uysal, Alistair Veitch
- **ex-SSP:** Liz Borowsky, Susie Go, Richard Golding, David Jacobson, Ted Romer, Chris Ruemmler, Mirjana Spasojevic
- **Others:**
  - Ed Grochowski (IBM Almaden)
  - David Nagle & Garth Gibson (Carnegie Mellon)

- **To learn more:**
  - [www.hpl.hp.com/SSP](http://www.hpl.hp.com/SSP)

References – workload characterization

- **Workload characterization**
  - [Ousterhout85], [Mogul87], [Baker91] – SOSP
  - [Miller91] – IEEE Mass Storage
  - [Ramakrishnan92], [Gribble98] – SIGMETRICS
  - [Caceres91], [Paxson94] – SIGCOMM
  - [Paxson97] – ACM Transactions on Networking
  - [Bates91] – VAX I/O Subsystems
  - [Ruemmler93], [McCanne93], [Roselli00] – USENIX
  - [Gomez98] – Workshop on Workload Characterization
  - [Hsu99] – UC Berkeley Tech Report
  - [Grimsrud95] – IEEE Transactions on Computers
  - [Touati91], [Eick96] – Software Practice & Experience
  - [Heath91], [Malony91] – IEEE Software
  - [Hibbard94] – IEEE Computer
  - [Aiken96] – Int’l Conference on Data Engineering
  - [Livny97] - SIGMOD
References – device modeling

♦ Device modeling
  – [Ruemmler93] - USENIX
  – [Worthington95], [Shriver97] – SIGMETRICS
  – [Pentakalos97] – Software Practice & Experience
  – [Thomasi94] – ICDE
  – [Menon97] – ICDCS

References – system design & allocation

♦ System (re)design and allocation
  – [Dowdy82] – ACM Computing Surveys
  – [Wolf89] – SIGMETRICS
  – [Pattipati90] – ICDCS
  – [Awerbuch93] – STOC
  – [Coffman84] – in Algorithm Design for Computer System Design
  – [Toyoda75] – Management Science
  – [DrexI88] – Computing
  – [Trick92] – Naval Research Logistics
References – monitoring & runtime

- Online monitoring
  - [Miller95] – IEEE Computer

- Runtime & distributed file system
  - [Lee96], [Gibson98] – ASPLOS
  - [Golding99] – Symp. On Reliable Distributed Systems
  - [Borowsky97] – Int’l Workshop on Quality of Service
  - [Bruno99], [Golubchik99] - IEEE Int’l Conf. on Multimedia Computing
  - [Wijayaratne00] – Multimedia Systems
  - [Gibson97] – SIGMETRICS
  - [deJonge93], [Anderson95], [Thekkath97] – SOSP
  - [Hartman99], [Amiri00] – ICDCS
  - [Howard88] – Transactions on Computer Systems

References – smart devices & availability

- Device intelligence
  - [Wilkes92] – USENIX Workshop on File Systems
  - [Cao94] – Transactions on Computer Systems
  - [Wang99] – OSDI
  - [Keeton98] – SIGMOD Record
  - [Riedel98] – VLDB
  - [Acharya98] – ASPLOS
  - [Uysal00] – HPCA
  - [Riedel00] – SIGMOD

- Describing manageability and availability
  - [Brown00] – USENIX
Sources for additional information

♦ Our web page – www.hpl.hp.com/SSP
♦ HP SureStore – www.enterprisestorage.hp.com
♦ Storage Network Industry Assoc. – www.snia.com
♦ Disk/Trend – www.disktrend.com
♦ IDC – www.idc.com
♦ IBM Storage – www.storage.ibm.com/technolo/grochows/grocho01.htm
♦ CMU Parallel Data Lab – www.pdl.cs.cmu.edu

♦ Tioga, The Holy Grail of Data Storage Management
♦ Farley, Building Storage Networks
♦ Gray & Reuter, Transaction Processing
♦ Bates, VAX I/O Subsystems: Optimizing Performance