zmanda
Open Source Backup

# Amanda
# The Open Source Backup Platform

UM-LUG (March 26, 2008)

Dustin J. Mitchell
dustin@zmanda.com

# About Me

- Amanda user for about 8 years
- UNIX sysadmin
- K-12 Teacher and IT manager
- FOSS contributor and author

- Storage Software Engineer at Zmanda, Inc.
  - One of the most active developers of Amanda
  - Very interested in cultivating the Amanda developer community

# Syllabus

1. Amanda's background and basic operation
2. Current development in Amanda
3. Opportunities to help out
4. My thoughts on open source development

Please stop me at any time for questions.

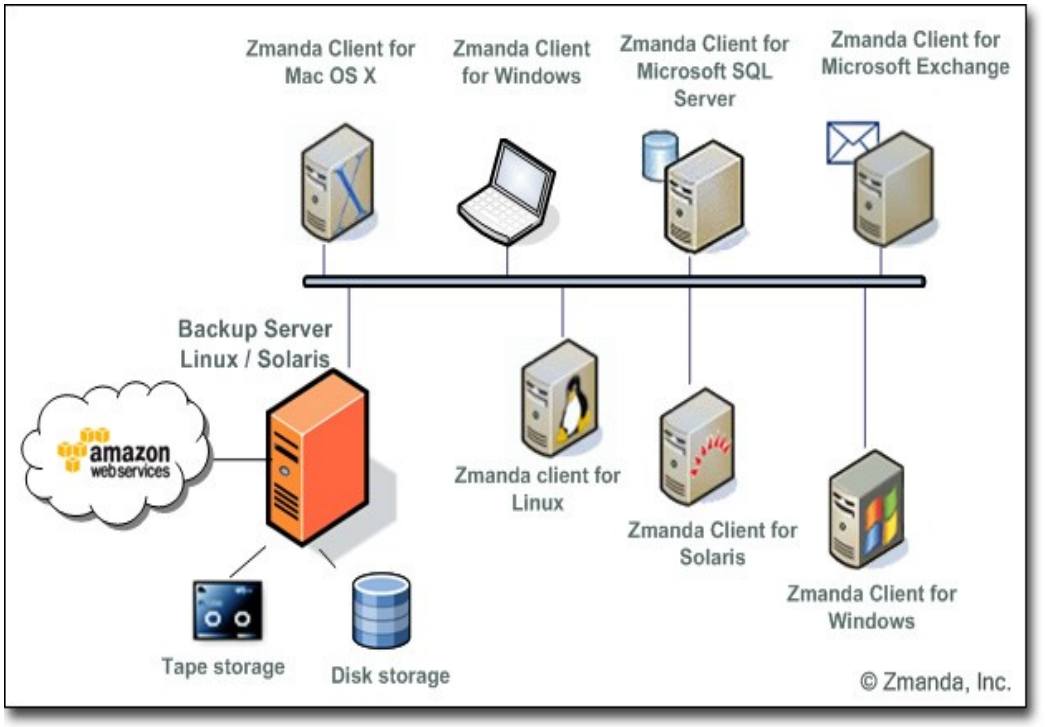
# About Amanda

- Performs LAN-based, multi-host backups
- Uses native client tools to perform backups and restores
- Supports network-based, file-by-file restores
- Unique backup planner gives consistent backup window
- Performs both disk-to-disk and disk-to-tape backups
- Battle tested (16 years) with large installed user base
- Has regular, backward-compatible release cycle

- Was originally written here at UM, by James da Silva and colleagues

# Amanda's Advantages

- Easy to use
- No proprietary data formats
- Scalable and heterogeneous
- Designed for backup to disk, tapes, storage grids and optical devices
- Secure
- Very active user community



# About Zmanda

- Provider of open source backup and recovery products and solutions
  - Amanda Enterprise Edition
  - ZRM-MySQL (MySQL backup and recovery)
  - ZMC (GUI interface)
  - Various Amanda plug-ins
    - Exchange client
    - Native windows client
    - MS SQL client (coming soon)
  - BackupPC (coming soon)
- Active developer and supporter of Open source backup and recovery projects
- Our products and support offered on an annual subscription basis
- We also offer professional services and solutions based on the enterprise products.
- We're hiring.

# Amanda Development

- Goals
  - Pluggable architecture
  - Support parallelism (multiple devices, multiple backup servers)
  - Append to tape
  - Support ILM techniques
    - move dumps between tapes (e.g., send all level 0's offsite weekly)
    - D2D2T
- Constraints
  - Backward compatibility
    - configuration
    - client/server communication
    - tape format and indices
  - Portability (runs on any UN*X, including Cygwin)
  - Minimal prerequisites, especially for the client
  - Manpower

Ideas

- FTP Device (Dreamhost, etc., or NAS)
- .Mac iDisk
- Interface directly with VTLs, other big iron?

Projects

- No devices actually implement locking yet
- Configuration of devices needs work
- Changer API is not well-integrated
- Add selectable mirroring and striping modes to RAIT

## Device API

Pluggable Interface to Backend Data-Storage Devices

**Goals**:

- Volume Append
- Delete Individual files from a volume
- Locking to avoid device contention
- Variable block sizes
- Device-specific properties (get and set)
  - Access modes (read/write, read-only, WORM, write-only)
  - Block Size (for reads and writes)
  - Device Capabilities

Examples

**Tape Device**:
All OS-supported linear tape devices (Amanda is not device-specific)

**RAIT Device**:
Redundant Array of Independent Tapes (can mirror any Device)

**S3 Device**:
Use Amazon S3 to store data offsite cheaply and reliably

**VFS Device**:
Treat disks as tapes (also known as 'vtapes')

**Optical Device**:
CD-ROM, DVD-ROM, etc.

zmanda
Open Source Backup

Implementation

- XML for extensible communication
- Well-defined set of application operations
- Abstract the notion of "files" into "user objects"
- Pre & Post scripts

Provides some interesting development problems

- Indexing byte positions after encryption and compression
- Backward-compatibility with existing Amanda clients

## Application API

### Pluggable Interface to Client-side Dump and Restore Applications

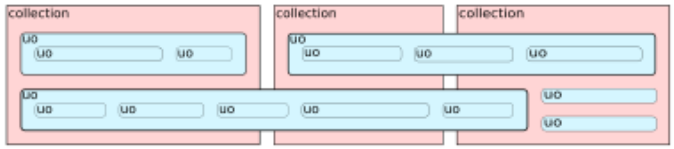
**Goals**:

- Send only required data to client on recovery
- Enable new applications
- Maintain backward compatibility
- Support native-tools restores
- Simplify implementation of new applications

### Data Organization

**User object**: restorable item (e.g., file, directory)
**Collection**: unit of output data from application

- User objects may contain other user objects
- Collections may contain many user objects.
- User objects may span multiple collections.




### Examples

**GNU Tar**:
**User Object**: file or directory
**Collection**: 512b tar block

**Native dump/restore**:
**User Object**: file or directory
**Collection**: entire dump datastream

**SQL Database**:
**User Object**: table
**Collection**: database dump

zmanda
Open Source Backup

Ideas

- Database engines (MySQL, Postgres, etc.)
- Schilly Tar (star)
- Cisco configuration
- Windows client
- Email applications
  - MS Exchange
  - Cyrus imapd
- NetApp's NDMP
- Source Code Repos (svn, git, hg)

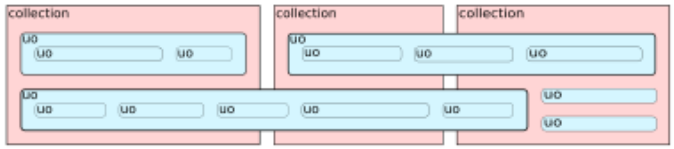
## Application API

Pluggable Interface to Client-side Dump and Restore Applications

**Goals**:
- Send only required data to client on recovery
- Enable new applications
- Maintain backward compatibility
- Support native-tools restores
- Simplify implementation of new applications

### Data Organization

**User object**: restorable item (e.g., file, directory)
**Collection**: unit of output data from application
- User objects may contain other user objects
- Collections may contain many user objects.
- User objects may span multiple collections.




### Examples

**GNU Tar**:
**User Object**: file or directory
**Collection**: 512b tar block

**Native dump/restore**:
**User Object**: file or directory
**Collection**: entire dump datastream

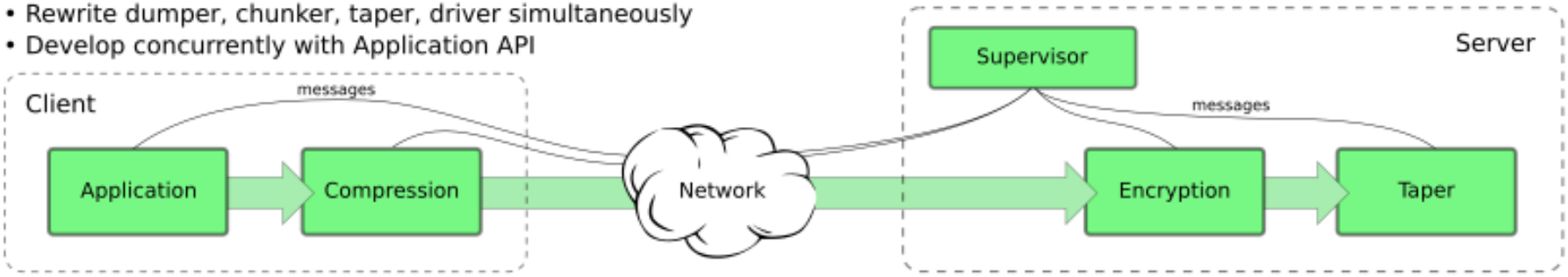
**SQL Database**:
**User Object**: table
**Collection**: database dump

# Transfer Architecture (XFA)

**Goals**:
- Improve bandwidth (minimize copies, maximize concurrency)
- Allow multiple concurrent transfers
- Enable data migration (e.g., D2D2T)
- Provide consistent 'toolkit' for dumps, restores, migrations
- Support Application API in sending minimal data for recovery

**Implementation**:
- Rewrite restore applications first
- Rewrite dumper, chunker, taper, driver simultaneously
- Develop concurrently with Application API

**Architecture**:
- A **transfer** is composed of source, filter, and dest **elements**
- Elements are joined efficiently: pipes, shmem, etc.
- Elements report progress via **messages** for indexing, status
- Datastreams are indexed in **slices**
- Filters perform encryption, compression, deduping, etc.
- All data-handling code is optimized C
- A transfer can span a network connection




The initial C implementation is committed, and I am building the Perl interface to it now.

There's lots of code to write, and lots of detailed design left to do.

Why?

- Encourage fine-grained patches from users
- Encourage new developers
- High-level operations are hard in C
- Scripting languages come with lots of utils
- We were basically doing a rewrite already

Why Perl?
(Python? Ruby? PHP? Shell?)

- Admins tend to know Perl
- Perl is a fairly low-level language
- Perl is more commonly installed on base systems

## Perl in the Core

New functionality will be written in Perl

**Goals:**

- Support other projects' large-scale changes
- Simplify use of XML, Unicode, etc.
- Write high-level code in a high-level language
- Encourage new contributors and developers
- Maintain backward compatibility

**Check it out!**
amanda.svn.sf.net/viewvc/amanda/amanda/branches/perl

**How?**

- Efficiency-critical components (XFA) will remain in C
- Existing libraries will be interfaced to Perl (using SWIG)
- Tools will be rewritten one-by-one in a gradual process

**Other Benefits:**

- Unit tests for all modules, tools (using Test::Harness)
- Cleanup and documentation of existing APIs

Example

```
use lib '@amperldir@';
use Amanda::Device qw( :ReadLabelStatusFlags );

sub try_read_label {
    my ($device_name) = @_;
    if (!$device_name) {
        return $READ_LABEL_STATUS_DEVICE_MISSING;
    }
    my $device = Amanda::Device->new($device_name);
    if (!$device) {
        return $READ_LABEL_STATUS_DEVICE_MISSING
             | $READ_LABEL_STATUS_DEVICE_ERROR;
    }
    $device->set_startup_properties_from_config();
    return $device->read_label();
}

my $flags = try_read_label($ARGV[1]);
print join("\n",
           ReadLabelStatusFlags_to_strings($flags));
print "\n";
```

# Seeking Filtered Data

A dump is essentially:

tar -cf - . | gzcat | aespipe | taper

We also gather information on which 'tar' blocks are required to restore each file.

**Q:** Given a filename, which blocks does the server need to read off tape to restore that file?

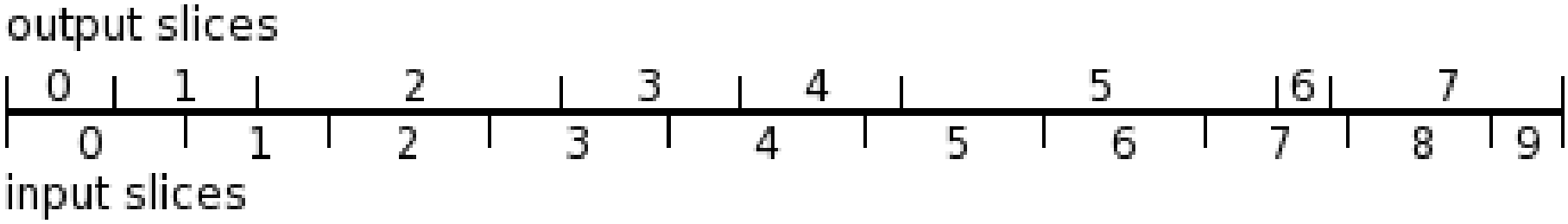
# Seeking Filtered Data

Let's look at the boundaries between transfer elements: a bytestream.

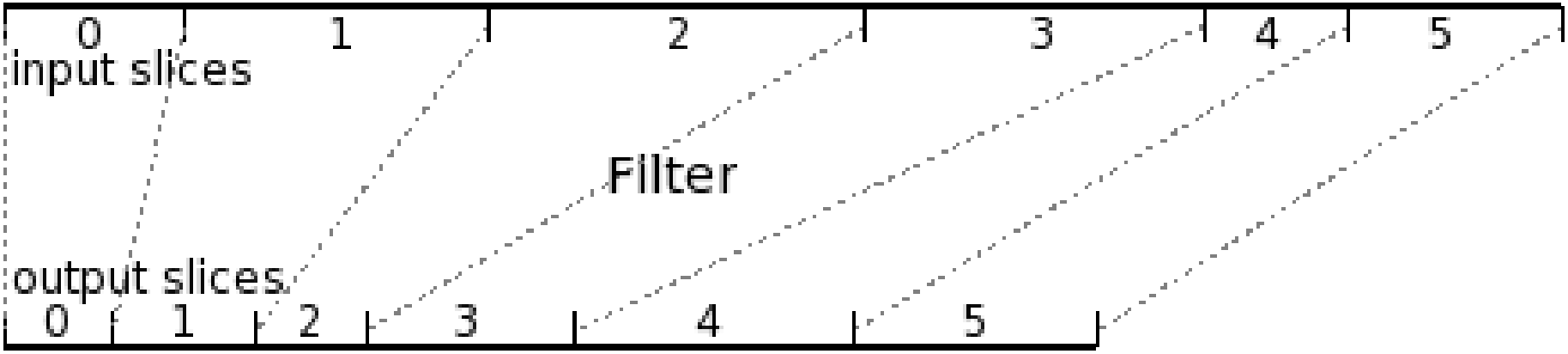
Upstream element divides the bytestream into "output slices" (a byte position interval)

Downstream element divides the bytestream into "input slices"

Slices do not necessarily correspond



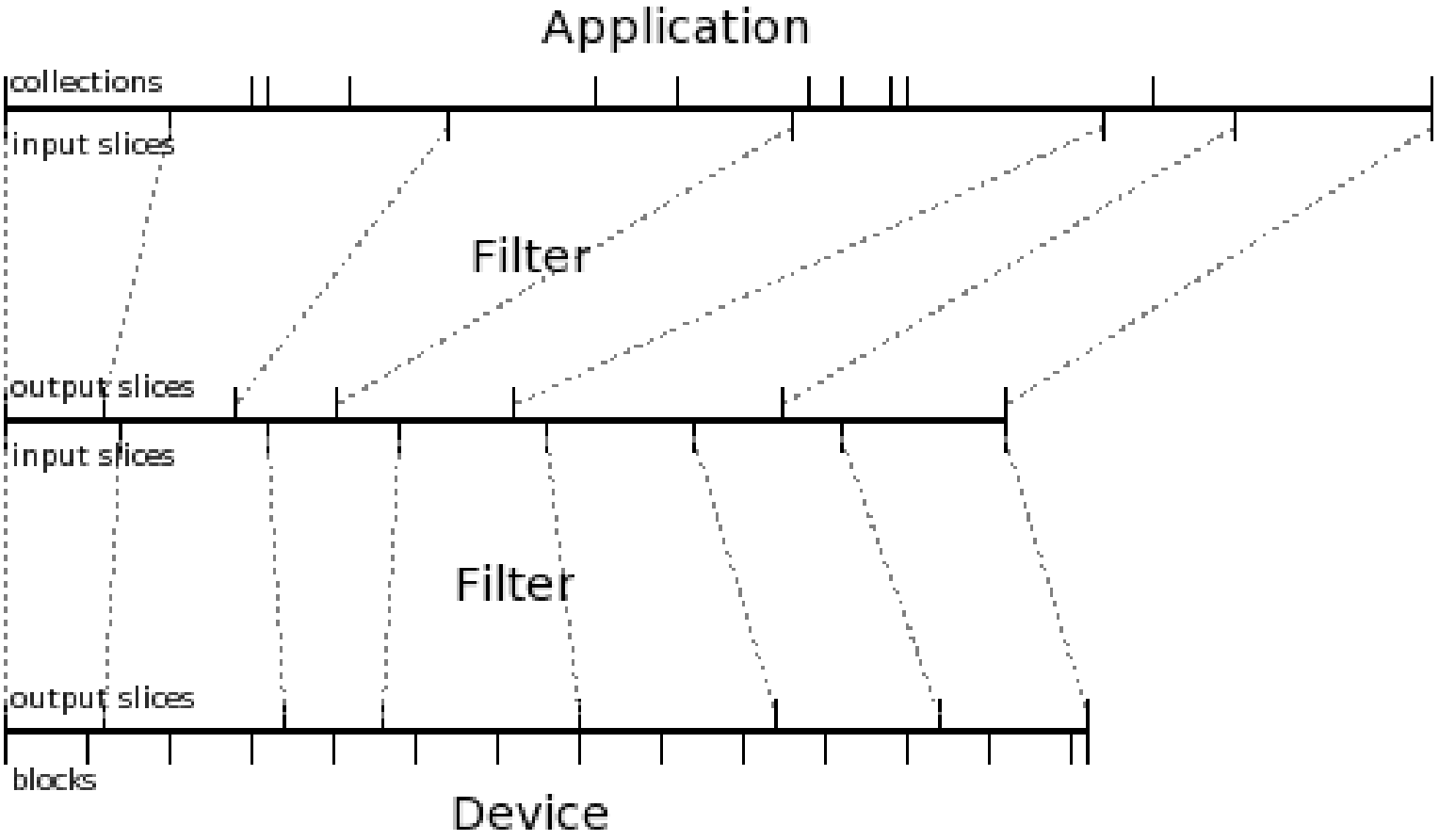
# Seeking Filtered Data



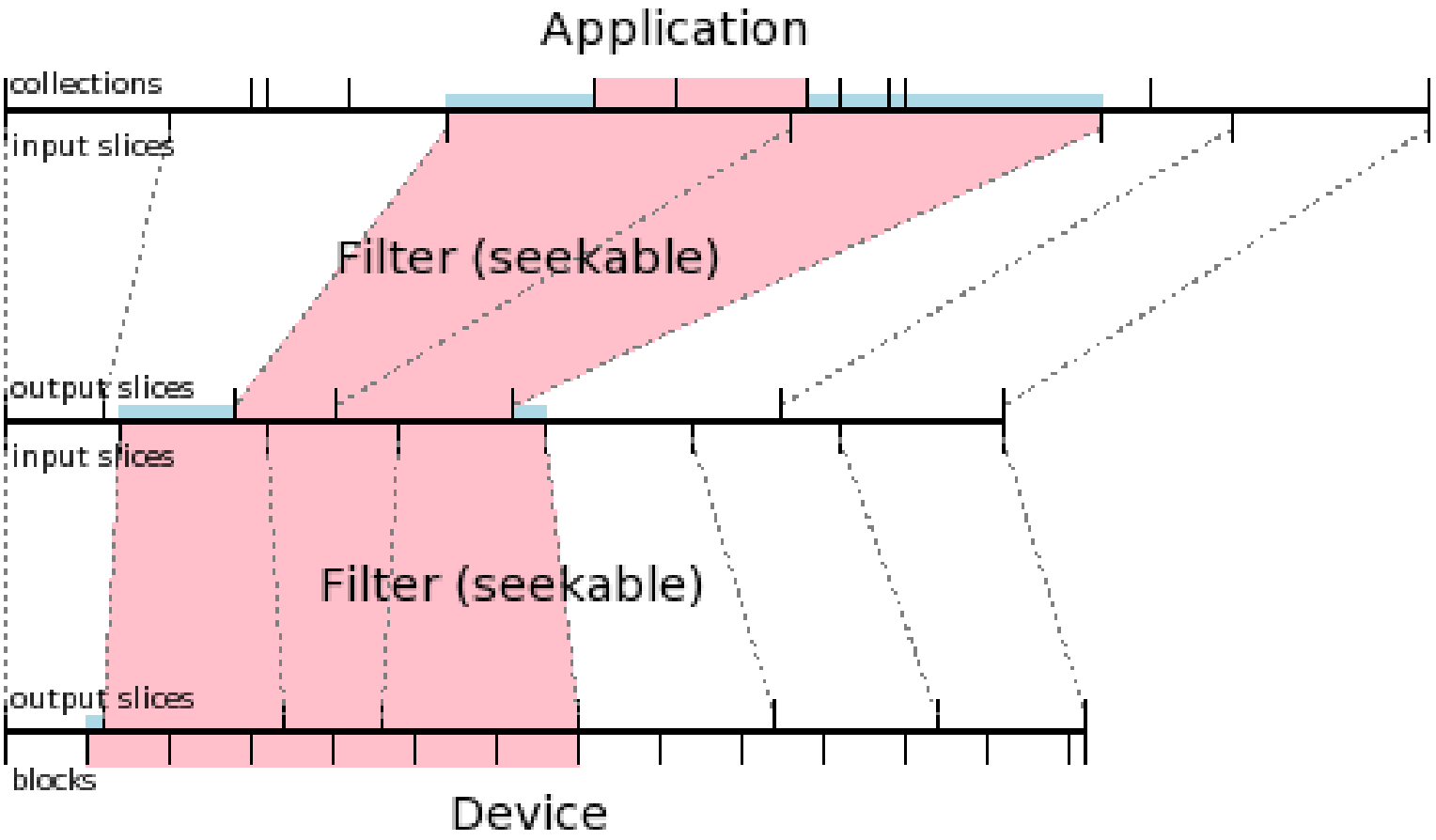

Note that input and output slices *correspond one-to-one*

*seekable filter* - can start at an arbitrary slice during restores
*catenary filter* - filter is distributive over concatenation

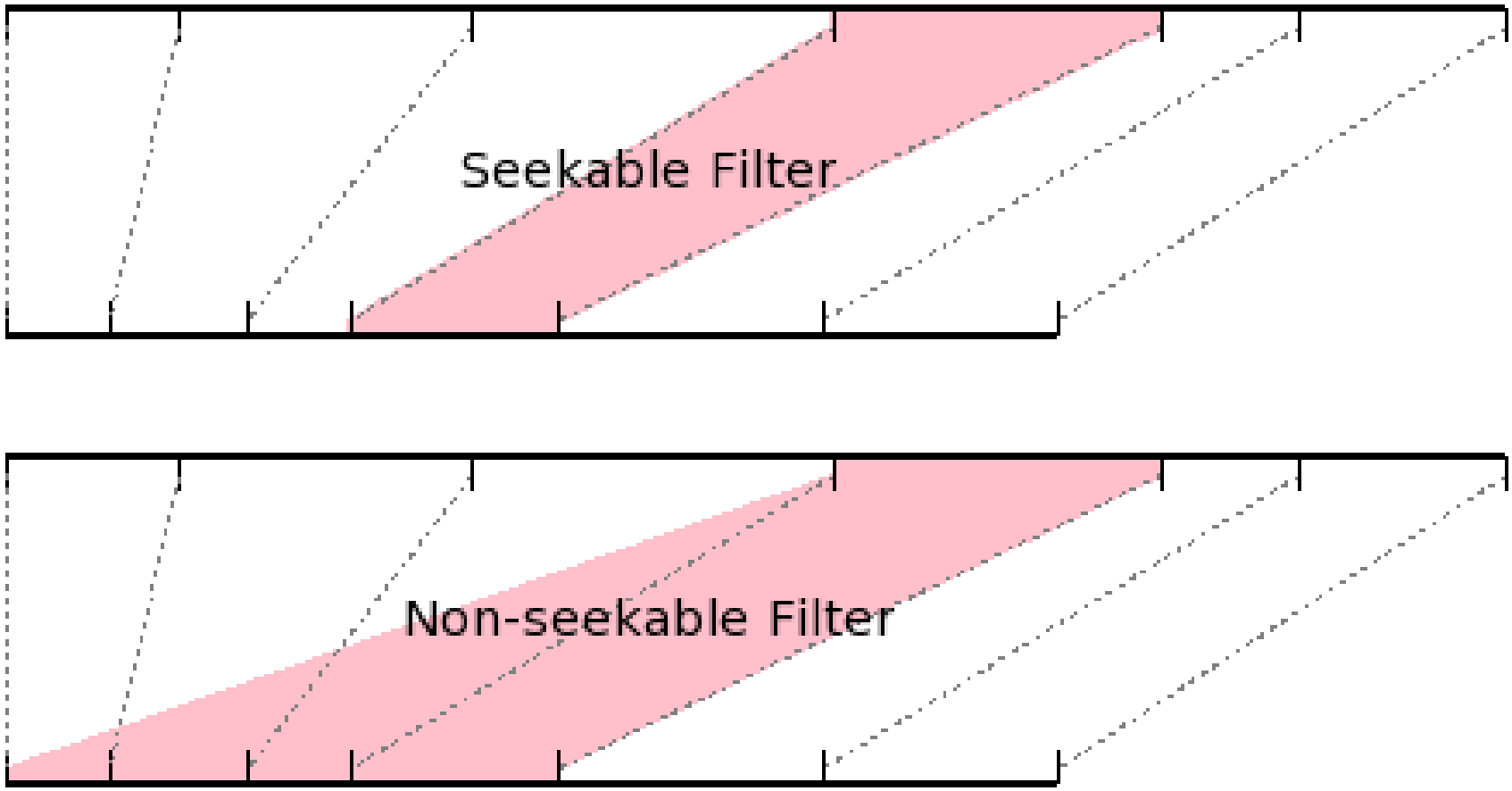
# Seeking Filtered Data



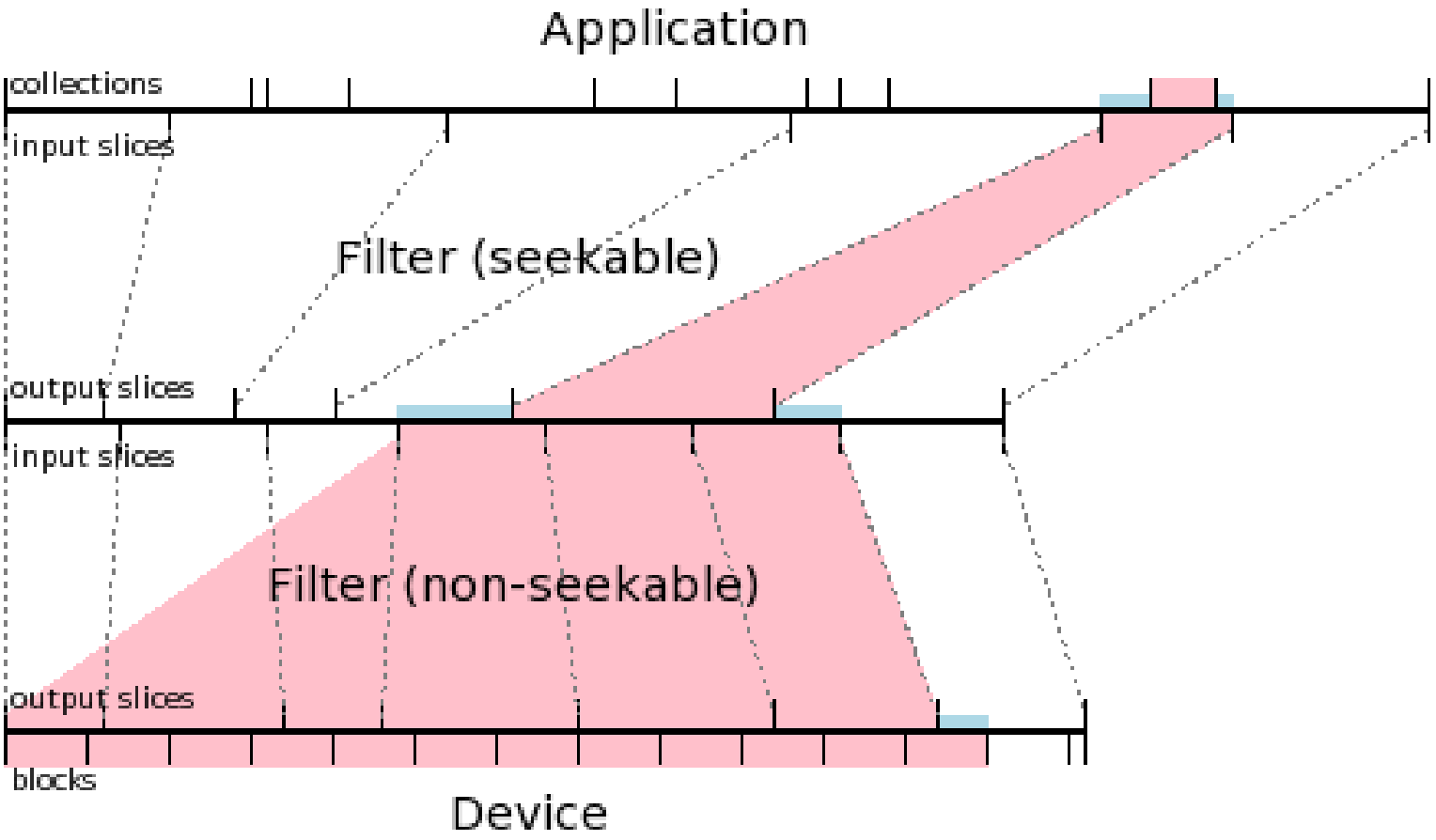
# Seeking Filtered Data



# Seeking Filtered Data



# Seeking Filtered Data



# Seeking Filtered Data

Seekable filters are required to allow quick restores.

Catenary filters are required to support more than one slice per bytestream.

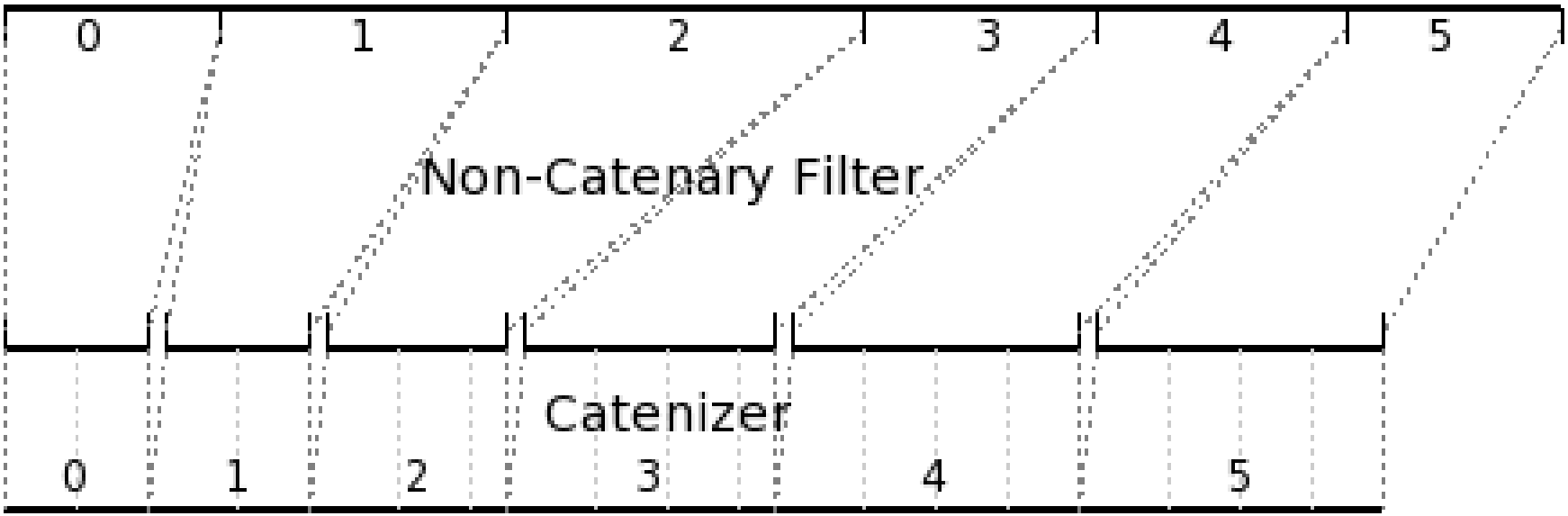
- bzip2: seekable, catenary
- gzip: non-seekable, catenary
- aespipe (or any encryption filter): non-seekable, non-catenary
  - Problematic for restores without indices

A non-seekable, catenary filter can be made seekable by re-invoking the filter for each slice, e.g., gzipping each slice independently.

So we need a way to make an arbitrary filter catenary...

# Seeking Filtered Data

*catenizer* embeds just enough information in the output bytestream to know when to restart the non-catenary filter when doing a restore, even when the index data is unavailable (native-tools restore, re-index operation)



# Join us!

- Become an Amanda user
  - Test new releases
  - Answer questions on the mailing lists, IRC
  - Improve Amanda's documentation
- Contribute patches
  - Fix a bug you've noticed, or add a feature you'd like to see
  - Complete a coding task (see "Tasks" on the wiki)
  - Amanda devs will help you with any problems
- Become an Amanda developer
  - Set your own goals for the project
  - Engage with other developers in design discussions
  - Become famous like Linus Torvalds or Larry Wall (maybe)
- Work for Zmanda
  - Get paid to write open-source code!

# Open Source Communities

# What is an open-source community?

- How do I "belong"?
- What makes a "strong" community?
- User community vs. developer community?
- How do communication mechanisms affect this?
  - IRC
  - Email
  - IM
  - Conferences (e.g., PyCon, ApacheCon)
- How can a community sustain its resources?
  - web hosting, wikis, mailing lists, subversion hosts, etc.
- Why do some projects have large development communities, while other communities are very small?

# Voice and power

Who gets to make the decisions?

- Main developer makes the call (Linus and his inner circle)
- "Rough consensus and working code" (IETF)
- Voting schemes (e.g., Apache Software Foundation)
- Relative power of individuals vs. companies (Zenoss, Zend, Zmanda)

Who takes the legal risks?
Who makes the contractual representations?

# Access and Exclusion

- Knowledge
  - How good is the documentation?
  - Where do newcomers find institutional knowledge?
  - Do I have to "speak the language"?
- Explicit barriers
  - Commit rights
    - Tigris.org requires a recommendation from another committer
  - Copyright assignment (e.g., FSF)
  - Tests and quizzes (Gentoo ebuild quiz)
- Developers' attitude to newcomers
  - Will the developers accept my patches?
  - Will I get help when I'm stuck?
- Threshold difficulty for entry
  - Are there "easy" tasks to demonstrate competency?

# Building a Development Community

My ideas:

- Lower barriers as much as possible
  - More accessible language (Perl)
  - Developer documentation (comments and wiki articles)
  - List of suggested "easy" tasks (on wiki)
- Welcoming attitude
  - Suggest opportunities for users to contribute (move from user community to developer community)
  - Offer help to anyone working on patches
- Speaking at LUGs, BUGs, conferences, etc.

What are your suggestions?