Lazy Greg's List of OVO Wishes

Greg Baker (gregb@ifost.org.au)

June 13, 2007

Contents

Introduction

I like to call myself an experienced consultant and instructor on HP's OpenView suite – close to 10 years now. When I'm pitching for a job I like to call myself "one of the leading OpenView experts in the Asia-Pacific region".

I've also worked at Google for a while, which is where I started to form crazy ideas about automated system and network management systems for *very* large networks.

I'm quite happy to implement some of these ideas for money (hint, hint!), but I'm also happy for them to be taken up by someone else in exchange for unlimited bragging rights ("see that feature – I suggested it!") and a warm glow of making the world a better place.

Some of these ideas aren't particularly coherent, and I'm writing this while teaching a class so it will be a bit disconnected. Ask me[1] if anything doesn't make sense and needs clarification.

1 log4j / log4c opcmsg appenders

`log4j` doesn't need much of an introduction. It's a nice way of being able to control where logs go – out to files, or out to a socket.

It would be nice if OVO shipped with Java and/or C libraries so that it could get `log4j` to send log messages out as `opcmsg` messages.

[1] `gregb@ifost.org.au`

And since the whole world doesn't use Java, doing the same for `log4c` and `log4perl` and so on might be nice.

2 Use a more Linux-like agent install process

While it's nice to have the consistent `opc_inst` script across all Unixes, it doesn't make sense on Linux.

OVO/U has an Apache installed; OVO/W modifies the IIS setup. We just need also to map a URI to the directory where the agent software is installed.

Then the install script becomes something like:

```
rpm --install http://mgtsvr/lxagt/agent.rpm
```

While we're at it, do a proper `.deb` format for Debian packages. Yes, I know that's not according to the LSB standard; yes I know that Debian has essentially zero impact on the market. But it would make more sense, and Ubuntu *will* be a big player.

For Debian/Ubuntu, the install script should be something like:

```
echo deb http://mgmtsvr/lxagt debs\
     > /etc/apt/sources.list
apt-get update
apt-get install ovoagent
```

It makes a big difference when the OVO server is patched; it makes it much easier to keep all the agent software versions in sync.

And, maybe it's worth shipping the agent with a static libc. Yes it's a bit wasteful not linking to the shared library version of it, but it's better to waste some disk and memory but always work and always install cleanly. Some story for libstdc++.

3 Cross-platform installs

- Unix OVO should be able to install to Windows boxes via `smbclient`.
- MS-Windows should be able to install to Unix boxes using SSH.

4 Inverted statistical thresholds and projected statistical thresholds

This is mainly of interest for disk usage, but it applies wherever we have a measurement of a limited resource.

I want to be able to create a threshold/measurement template/policy as usual, but specify a rule that triggers "when the value gets within 4.5 standard deviations of 100%" (another way of saying roughly the same thing is "when the probability of the next data point being at 100% gets greater than 1 chance in a million").

This makes filesystem capacity monitoring easy – filesystems which do not change in size much (e.g. `/usr` on HP-UX) will have a very small standard deviation, so could sit at 95% full without that being an alarm-worthy problem. A filesystem which does change in size rapidly will have a large standard deviation, and would alarm long before 95% full.

To do this will require (I think) storing sum-of-squares data as well as the data values themselves, so we can kill a second problem at the same time – filesystem which grow with time. I would like to specify a rule that triggers "when there is a 1% chance that a data point in the next 2 weeks will be 100". This is just a linear regression on the data, and projecting it out.

5 Extra Performance Agent metric

In an era of disk arrays which have cache, measuring the number of bytes in/out to a LUN tells us very little, and the number of reads or writes is not much help either. Perhaps they are all landing in array

cache (and so all is good) or perhaps none of them are (not good).

So here's the metric that would help... in the last `$TIMEINTERVAL`, what percentage of blocks in this filesystem were modified?

6 Policy group inheritance

Wouldn't it be nice to set a default message group for a whole policy group? Or set other kinds of defaults?

(Actually, I'm not fussed about this one, but a customer requested it very politely.)

7 Shared IMAP Interface for viewing messages

Some staff don't have the screen space to run an extra program. Or they forget to start up `opc`. Or whatever. But they are likely to have their email client open.

So the OVO server should run an IMAP/IMAPS server, but instead of presenting internet email messages, it presents active OVO messages in a very email-like way.

e.g. a critical message from `syd42` (application name="oracle", object name="production", message text="Out of table space") would be presented as if it were an email:

```
From: oracle!production
To: syd42
Subject: Out of table space
Priority: high

Out of table space.
```

Or you could play games with IMAP folders being the different message groups. When a user deletes this pseudo-email that either acknowledges the message or just hides it for this user, depending on who the user's account has been set up.

The advantage of this approach is that

- Managers who don't know anything other than Outlook can still have some idea of what is going on.
- Sysadmin geeks who refuse to use anything other than *insert obscure operating system here* can use their favourite mail client to get notified about things.
- It becomes possible to drag a message to become a To-Do item.

When I'm feeling very silly, I also wonder about a jabber interface to the unacknowledged messages store.

8 Consolidated command broadcasting output

The current output from a broadcasted command (i.e. the one program run on many computers at the same time) gets unwieldy very quickly. No-one has the patience to read through the same output text 50 times; so that limits commands to less than 50 machines, unless the program has almost no output.

Here's what I'd like to see:

```
ivm211, ivm212, ivm213 produced:
   Permission denied.
ivm221, ivm222, ivm223, ivm224, ivm225
   Sending glub daemon a TERM signal.
   Sending glub daemon a KILL signal.
   Starting glub daemon.
   Glub daemon started                | ALL EXCEPT FOR ivm22
   Glub daemon running; config loaded.| ALL EXCEPT FOR ivm22
   Glub daemon cannot start. No config| ivm225
```

Maybe colour highlighting would work better.

Anyway, I've done some thinking about algorithms to generate this kind of output. Ping me if you're interested.

9 Deduce correlations

Every night OVO should run a job searching for pairs of events that regularly occur together.

If every time event A happens, an event B happens in the next 30 minutes, that suggests that A causes B, or that B is just another test for A.

If they have the same service ID, it suggests a redundant message (and we could let the administrator know somehow that this would be a good ECS correlation to create to suppress one or the other).

If they have different service IDs, it suggests that there should be a "propagate most critical" calculation rule on one of them.

10 Bayesian rules in logfile policies/templates

The point of the message browser is to highlight the important events going on in the customer's systems so that operators and sysadmins can respond. They know what messages are important and which are not.

Here is my shortlist of words which indicate – with very high probability – a message which a sysadmin will care about if they appear in a line in a log file.

- No as a word on its own
- not
- didn't, couldn't, isn't
- denied, failed
- out of, too

So why not start with that collection, and then let the operator indicate how well it is working? It works pretty well with email spam. . . .

11 Find logfiles automatically

I was so disappointed with log file discovery in OVOW 7.5 – I was expecting it to appeal to my lazy side.

Here's how an agent can identify every log file on a system:

1. Look at `/etc/syslogd.conf` and mark anything listed there as being a real log file.
2. Look at `/etc/logrotate*` and figure out any file names that are just renames and compressions of real log files. Mark the compressed/renamed files as boring, and remember the real log files.
3. Traverse all local filesystems looking for ordinary files (i.e. not directories, not fifos, etc.), which we haven't already marked as being a log file, boring, a configuration file an executable or a shared library.
   - Store checksums and sizes of all such files which are not directories, not fifos, not marked as being log files, not marked as boring, not marked as a configuration file, and not marked as an executable or shared library.
   - Mark any executable programs and shared libraries as being such.
   - Mark files ending in `.png`, `.gif`, `.jpg`, `.html` etc. – which happen to be in the format their filename suggests – as being boring.
4. Wait a while (auto adjust the interval depending on how many new log files we've been discovering – slow down and wait much longer when we haven't seen a new log file in a long time).
5. Traverse the filesystem again, and consider all files that have increased in size. Checksum just the first N bytes of those files (where N is the size it was when we last looked at it), and see if the checksum is the same.

- If it is the same, then mark this as a log file.
- If it is not the same, mark this file as boring.
- If a file gets smaller, look for a file in the same directory with the same name, but with a number or date appended, and try checking that file's first N bytes.
- If the checksum is different, and there are non-linguistic combinations (e.g. two punctuation characters together beside a non-ascii character), then it's probably a data file. Mark it as boring.
- If the checksum is different, but it is a plain text file, then mark it as a configuration file.

6. Go back to step 5.

OVO could ship with a subagent that did all the above, and a policy which uses Bayesian rules (from idea 10 on page 4). This would produce a genuine "works-right-now" solution – you could install it and do *no other configuration* but still end up with a useful systems management solution.

11.1 Configuration file watcher

If idea 11 (page 4) is implemented, then as a side-effect, configuration files get automatically identified. A separate subagent could monitor configuration files for changes and do several things when a change is noticed:

- Log the change in a version control system (e.g. subversion). SuSE Linux has a `cfg2scm` program for this, for example.
- Send a message to the browser.
- Log some kind of "match-me" ticket in a change control system, with an alert generated if this configuration change can't be connected to a change control request number.

11.2 Executables and shared libraries watcher

Taking idea 11 (page 4) a bit further, why not monitor executables and shared libraries for changes?

When a change occurs:

- Bundle all the changed files (including configuration ones) together in an RDP (remote deployment pack) object, so that it could be repeated precisely on other machines.
- Make sure that there is a change control ticket associated with the job to apply patch.