Zkk, The Infinite Improbability parallel zypper download shim for OpenSuse

License Agreement:

This software is provided to OpenSuse users in hope that it will make their experience with OpenSuse a more positive one.

zkk, ZK_downLoader nor the author is in any way affiliated with Suse or Opensuse

If you would like to support my work

Paypal: zkk.opensuse@gmail.com

When running zypper you do want to install/update the libregina3-devel-3.9.1, hopefully some kind sole will update Regina REXX in the openSUSE repos.

Documentation:
The REXX scripts go in /bin/:
zkk
ZK_downLoader
Features:

- zkk gets the patch listing from zypper.
- Parallel patch downloading up to 42 at one time configurable, it also checks to see if the rpm signatures are good, provides SHA-256, SHA-1 and MD5 hash. I thought the SHA-512 is a little exuberant, and it is just more data to move. You can change the number of parallel downloads using aria2 while the program is running and it will adjust to to new setting up or down without restarting. (My machines can not spawn the download program fast enough to get to that number, or maintain the logs fast enough, there is always a constraint.)
- It builds a parallel zkk repo files in /etc/zkk_download_shim/repos.d these repo files support multiple download URLs which can be cycled through, up to 256. Average download speed is tracked so the fastest ones will be at the top on the list. **YES**, you can have multiple sources for the same patch file if the download fails the program will automatically pick a new one, no more pushing retry. Each entry (row) in the file contains a URL these can individually be enabled and disabled.
- In the zkk.conf file you can control how many times aria2 retries, how many times ZK_downLoader will try other repos (URLs), how log it waits before retrying again, one download does not block another.
- Tracked by repo URL enabled, Average download speed, good downloads, failed downloads, failure date and time, these will roll off as new ones are added.
- Logging in /var/log/zkk_download_shim, one log for all, with lines timestamped and with PID, program and repo. zkk does have a problem keeping up with writing the logs, this problem is non-blocking.
- Before each dispatch cycle checks memory, diskspace free, and how many ZK_downLoader are running configurable in /etc/zkk_download_shim/zkk.conf

Dependencies:
Internet

/etc/zkk_download_shim/zkk.conf (zkk, will create one if it does not exist)

Aria2, aria2 is a lightweight multi-protocol & multi-source command-line download utility. It supports HTTP/HTTPS, FTP, SFTP, BitTorrent and Metalink. aria2 can be manipulated via built-in JSON-RPC and XML-RPC interfaces.
https://aria2.github.io/

Regina REXX 3.9.6 (Rexx (Restructured Extended Executor) is a programming language that can be interpreted or compiled. It was developed at IBM by Mike Cowlishaw.)
https://regina-rexx.sourceforge.io/
rxstack (Included with Regina REXX)

Packages dependencies for Regina REXX libncurses5 libreadline6, installed by zypper

Installation of zkk:

The zkk package contains 4 files:
zkk.rex (Main and download control loop, Thread)
ZK_downLoader.rex (Performs downloads, rpm checksig and hashes)
zkk_sed_script.sh (Edits the scripts so they find regina)
zkk.conf (Configuration file)
zkk.odt (This documentation)

Installation of prerequisites:
1. Install with zypper as root:
zypper install libncurses5 libreadline6 aria2

Download from the Regina REXX site in the 3.9.6 folder and run, you do not want the dev ones.
rpm --install libregina3-3.9.6-240341.1-x86_64-RHEL-7.rpm regina-rexx-3.9.6-240341.1-x86_64-RHEL-7.rpm

run the script zkk_sed_script.sh, the script checks to see that regina REXX is installed.
This will change the first line in the *.rex scripts so the script will find regina make them executable and output them to zkk and ZK_downLoader.

Copy zkk and ZK_downLoader to the /usr/bin folder

If you get something close to the bash error: zkk: regina: bad interpreter: No such file or directory run which regina and check line 1 of the script, for where regina is located in the path.

You can also copy the zkk.conf file to the /etc/zkk_download_shim/ and edit it to your preferences if you do not the script will create it for you.

The program creates the following folders and files the rights are 644:
/etc/zkk_download_shim/
/etc/zkk_download_shim/zkk.conf
/etc/zkk_download_shim/repos.d/
/etc/zkk_download_shim/repos.d/*, mirror of the systems repo files.
/tmp/* used for zypper output
/var/cache/zypp/packages/*/* used to place validated *.rpm files for zypper
/var/log/zkk_download_shim/zkk(date timestamp).log these are large files you are encourage to remove them when no longer needed, or set up logrotate to help manage files under /var/log

Running zkk:
Open a bash console and su to root
#rxstack -d (Starts rxstack as a daemon)
#zkk
Maybe
 # rxqueue /list (To see the queues inside rxstack)
 # rxstack -k (Removes rxstack as a daemon)
 # zypper [d]up

Configuration file documentation:
/etc/zkk_download_shim/zkk.conf:

Config file for zypper download shim
Comment the first column starts with a #

Cleanup_Temp_Files_After_Run Y (Y/N)
Remove /tmp/Zypper_List_Updates.txt file at the end of run
Min_Memory_MB 1024 (Minimum is 32, no maximum, dynamic)
Minimum available memory in MegaBytes (RAM) for running the program or starting another ZK_downLoader, values are reported by free -m
Min_Disk_MB 1024 (Minimum is 512, no maximum, dynamic)
Minimum available disk space in MegaBytes for the Working_Directory (default is /tmp) for running the program or starting another ZK_downLoader, values are reported by df
Repo_Refresh_Retries_Int 3 (Minimum is 2, no maximum)
The number of times zkk tries to call *zypper refresh* and get a notification that all the repos are up to date.
Valid_Protocols http,https (http, https, ftp and sftp)
Valid Protocols for aria2 to use http, https, ftp and sftp separated by commas and no / (slashes) or : (colons). If the protocol is not listed the URL will be marked invalid.
ZK_downLoader_Debug 0 or 1 (Minimum is 0, Maximum 1, Integer, dynamic)
I put this into filter lines pushed to the stack from ZK_downLoader. I have/had some performance problems with processing lines from ZK_downLoader in zkk. In ZK_downLoader it really does not matter they are just off running on there own. But pushing everything back zkk does not process the lines fast enough so it does not keep up with creating ZK_downLoader processes.
Max_Concurrent_Downloads_Int 8 (Minimum is 1, Maximum 42, dynamic)
Number of simultaneous ZK_downLoader which can be running at the same time. This seems to run a little hot, set to 8, I have seen it up to 12, the important thing is it does not run away.
Connect_Timeout_Sec 15 (Minimum is 5, Maximum 300, dynamic)
This is an Aria2 variable which is passed through. Here is the aria2 information, “Set the connect timeout in seconds to establish connection to HTTP/FTP/proxy server. After the connection is established, this option makes no effect and --timeout option is used”.
URL_Retry_Wait_Sec 15 (Minimum is 1, Maximum 300, dynamic)
After a download has failed either aria2 or the ZK_downLoader loop (See technical documentation below) How log to wait before trying again. If aria2 the same URL if ZK_downLoader maybe a different URL
Lowest_Speed_Limit_KB 1 (Minimum is 1 Maximum 1024, dynamic)
This is an Aria2 variable which is passed through, The value expects just an integer, ZK_downLoader adds the K. Here is the aria2 information, “Close connection if download speed is lower than or equal to this value(bytes per sec). `0` means aria2 does not have a lowest speed limit. You can append `K` or `M` (1K = 1024, 1M = 1024K).”.
Max_Retries_Per_Site_Int 2 (Minimum is 1 Maximum 16, dynamic)
This is an Aria2 variable which is passed through. Here is the aria2 information, “Set number of tries. `0` means unlimited.”. If aria2 download fails it will try the same URL again this many times.
Max_Total_Retries_Per_Loop_Int 2 (Minimum is 1 Maximum 16, dynamic)
If aria2 fails in the download how many times will ZK_downLoader try to run aria2 again and download the patch, if there are multiple URLs available it will pick a different one.

Repo_Sites_Int 2 (Minimum is 0 Maximum 256, dynamic)
This controls how many URLs ZK_downLoader uses when the zkk repo file is read in all valid/enabled URLs are sorted descending by Average download speed, Repo_Sites_Int tells how far to go down the list to use from the top. So the fastest is first. Ie use the top 2, in this example. If it is set larger then the number you have enabled in the file it defaults to the number in the file.

```
cat openSUSE:repo-oss.repo
Enabled
  Average Download Success Failed  URL                                                          Date / Time Failure
1 0000000000120832 0000002 0000000 http://cdn.opensuse.org/tumbleweed/repo/oss
1 0000000000881236 0000013 0000001 http://opensuse.ipacct.com/opensuse/tumbleweed/repo/oss 2024-06-13 15:50:30
1 0000000000000000 0000000 0000000 https://provo-mirror.opensuse.org/tumbleweed/repo/oss
0 0000000000000000 0000000 0000000 https://ftp.cc.uoc.gr/mirrors/linux/opensuse/opensuse/tumbleweed/repo/oss
```

Work_Directory /tmp (default)
The work directory is where zypper writes out the package update list.
Cache_Directory /var/cache/zypp/packages/ (default)
The directory hive where ZK_downLoader writes the files for zypper to use.
Repo_Directory /etc/zypp/repos.d
The directory where zkk reads the zypper repo information from. zkk does not modify these files.
Run_zypper_After_Download N/Y default is N
Run zypper after download is complete, information is sent both to the screen and the log file.
Run_zypper_with_No_Confirm N/Y default is N
Run zypper with no confirm after download if complete, information is sent both to the screen and the log file. Both Run_zypper_After_Download and Run_zypper_with_No_Confirm need to be set to Y
Run_zypper_option up or dup (For Leap it is set to up, for Tumbleweed it is set to dup)
How to run zypper after download.
Write_Zkk_Repos_Every_Sec 300 (Minimum is 30 Maximum 1200)
This tells zkk how often chronologically to write out the files in /etc/zkk_download_shim/repos.d. You do not want this to run to often since it pauses running new ZK_downLoader, during the update.
Write_Zkk_Repos_Every_Calls 200 (Minimum is 100 Maximum 2000)
This tells zkk Main how often in calls made to ZK_downLoader to write out the files in /etc/zkk_download_shim/repos.d You do not want this to run to often since it pauses running new ZK_downLoader, during the update.

The dynamic variables can be updated while the programs runs and will be picked up. Basically if it runs in the zkk thread or ZK_downLoader.

System Technical Documentation:
The major problem with writing parallel code is keeping track and logging the information. Parallel code runs into file locking and sync, which are a big problem, one program can not read to the same file as it is being written. Regina REXX has this clever thing called rxstack so all the children can push the information there along with their PID and it can be sorted out and put into a single log file. So problems can be tracked down. When writing a program setup your logging first.

Progam Outline:
You need to run *rxstack -d*, to start the REXX stack. To kill rxstack, *rxstack -k*, To see the queues in rxstack, *rxqueue /list*.

This program runs as root. It does remove some files in /tmp (default) and /etc/zkk_download_shim/repos.d

zkk is the parent it first checks for dependencies

zkk is designed to run in a bash console, it starts writing to the screen and when sets up logging it switches to a log file located in /var/log/zkk_download_shim it has date and time stamps the log files will build up.

It creates needed directories and files, you are informed of this in the log file. It sets permissions for the directories and files to 644.

zkk then runs Init and reads from the configuration file /etc/zkk_download_shim/zkk.conf, if it does not exist it will be created for you, and it can be edited, the file is space delimited.

During Init zkk reads the **zypper** repo files in /etc/zypp/repos.d (These are the zypper repos), which contain one baseurl) I set up a parallel structure in /etc/zkk_download_shim/repos.d (These are the zkk repo files), These are laid out:
Enabled, Average download speed, Successful downloads, Failed downloads, BaseURL, Date failed Time failed, space delimited. All one one line for each URL, you can have as many lines as memory will support. Each repo line can be enabled or disabled individually. **If the baseurl does not contain tumbleweed or (leap with a release version) it is disabled,** zkk will let zypper pick those patches up. While reading in the zypper files and setting up the zkk files zkk creates a variable box called Repo_Box), wow that's original Repo_Box.1.X.Y contains information from the zypper files. Repo_Box.2.X.Y contains information from the zkk repo files. zkk uses Repo_Box.1.X.Y as pointers over into Repo_Box.2.X.Y, like filename, and repo name, it also gets its first baseurl from the zypper repo files.

zkk also kills packagekitd, then calls *zypper refresh* and *zypper list-updates > /tmp/Zypper_List_Updates.txt* and writes the information in the work directory (default is /tmp.

Ok, now we have all the information to begin downloading files. Here I call a Regina REXX fork() splitting zkk in two, a log processing loop main and a download loop thread.
There are zkk main running this handles logging and running zypper after all the files are downloaded, there is zkk thread which is the loop which spawns the ZK_downLoader(s) up to 42 these call aria2, rpm, sha256sum, sha1sum and md5sum. The zkk thread and ZK_downLoader(s) push their log entries to rxstack queues for zkk main to pick up and write out, zkk main works hard.
So at the 30,000 foot view we have.

Zkk Main, Thread and ZK_downLoader tasks

zkk Main, 1 Running	**zkk Thread, 1 Running**	**ZK_downLoader**
Reads queues	Keeps track of:	**1 to 42 Running**
Writes to logs	How many ZK_downLoaders	Downloads files using aria2
Keeping updating the zkk repo	are running	Uses rpm -checksig
entries and files	Free memory, Free disk	Hashes files
Can on exit call zypper	Which file to download.	
	Pushes logging messages to zkk	Pushes logging messages to zkk
	Main via the queue	Main via the queue Z(PID)
	Thread_Log_ZKK	

Now we go through a while loop and read line by line the Zypper_List_Updates.txt, checking available RAM and disk space how many downloads are running and if we are not at max we dispatch another one (I made a change for performance it does this before each set, it takes the count of ZK_downLoaders and then dispatches a set in a for loop), until we hit a max RAM disk or downloads.
ZK_downLoader is called in this format:

#_of_calls	repo_filename	arch	filename_to_download
4	openSUSE:repo-oss	noarch	adwaita-icon-theme-46.2-1.1.noarch.rpm

ZK_downLoader, does not write out logs the information is pushed to the rxstack. Logging will be covered later along with the rest of zkk.
ZK_downLoader can and should have many of them running at once. The #_of_calls field is for each repo file and can be used for multiple repo entries to step between them so if you have 2 or more URLs in your zkk repo file step through you can do this.

Let’s talk a bit about removing files: Yes, I do not like to do it either, there are two procedures which delete files: Remove_File and Remove_Working_Dir_File, before calling to delete the file zkk checks for the following in the filename * <space> and a minimum length of 8 characters the includes the directory path. Each time rm is called it logs the file it is removing. I ran into a little special thing with aria2 and filenames so I wrote Remove_Working_Dir_File this will remove all files with a .1 .2 .3 … .9999 in the file name it first does the same * <space> and a minimum length of 8 characters and adds a .* just before the .rpm at the end. Downloaded files written to the cache directory by ZK_downLoader.

ZK_downLoader, reads the parameters and runs Init reading the same /etc/zkk_download_shim/zkk.conf file.

1. Is the file already in the /var/cache/zypp/packages/ area passes rpm -checksig and does not need to be downloaded again? If the file fails rpm -checksig the rpm file is removed and re-downloaded. (zkk does not remove *.rpm files it would not overwrite).

ZK_downLoader now reads the zkk repo file in /etc/zkk_download_shim/repo.d after it reads in all the **ENABLED** lines it sorts the stem (array) descending so the fastest sites are at the top.

ZK_downLoader, does modulo arithmetic to find which repo line to pick and builds the call to aria2 and dispatches it and looks for “(OK):download completed”.
if ZK_downLoader sees “(OK):download completed” the program runs rpm -checksig (we do not want bad data), sha256sum, sha1sum and md5sum and exits.

if any of the above fails the file is removed and a counter (Cnt_Total_Retries_Per_Loop_Int) is incremented and if available, a different site is used until it runs out of retries.

The logging information is all being pushed into a queue area in rxstack, starting with a Z and then the program's PID, many ZK_downLoader can be running at the same time. You can list the queues by running
rxqueue /list at a bash shell while the program is running. Now the record keeping part of the documentation.

Logging and record keeping:
So now we will cover the logging information, log information is stored in /var/log/zkk_download_shim/zkk_(Date)_HH_MM_SS.log, the lines:

```
Date      Time           PID Program Module   Message
20240613 17:03:46.470378 1941 zkk     Main     Checking: Regina REXX found in:  /usr/local/bin/regina
```

zkk can keep up with 2 download streams YMMV, I re-wrote the program 3 times to get around this, no luck, above that it falls behind, at least on my test computer, it is running as fast as it can. It takes about a half second to process and remove a queue. If there are 10 second pauses between downloads, like I experience sometimes with zypper, zkk can easily keep up with it and you could just expand the number of parallel ZK_downLoader programs. The logs will get processed after the downloads are completed. You can list the queue by running rxqueue /list, zkk just stores them as different queues in the stack area, it is all non-blocking until zypper needs to be ran. zkk, writes out the log file covered above, so zkk Main writes out log messages from zkk Thread and ZK_downLoader, the zkk Thread pushes to the queue Thread_Log_ZKK, the ZK_downLoader pushes to Z(PID)

First an aside on REXX stems (arrays) REXX_Stem.0 tells you how many items there are in the REXX_Stem.

```
do J = 1 to  REXX_Stem.0
      say REXX_Stem.J
end /* do J */
```

Will write out all items in the REXX_Stem.

ZK_downLoader, after it is spawned from zkk, thread it waits till it finds it PID in the system, and then creates a queue with its PID number with a Z added to the front of it, then set the program to write to this queue. zkk main then waits for the PID to be gone from the system before removing the items from the queue and then removing the queue. The other logging queue is Thread_Log_ZKK, this queue is where zkk thread pushes all its information. Items from the Thread_Log_ZKK queue are removed each time it goes through a loop. So remove all items from Thread_Log_ZKK queue then go through all the Z(PID) queues and remove the Z(PID) queues which have finished then repeat.

As items are removed from the Z(PID) queues items are picked like the repo, URL ZK_downLoader was using, download speed, success, failure. These are updated in the Repo_Box.2.I.J memory area as the program runs and then written to the zkk repo files on (I am trying to be clear about which repo files are being updated) disk as controlled by the zkk.conf (either by time or by call) file. After they are written to disk, a newly run ZK_downLoader picks up the changes, the enabled, fastest download speed URL is sorted to the top.

When after a while either time or number of downloads zkk Main will request zkk Thread to pause creating new ZK_downLoader so that the zkk repo files can be updated the programs use rxstack queues to communicate.

After the download and writing the log file is complete.
After the download, zkk gets most of the packages, in one of the tests it downloaded 2191 of 2289.
After this there are 3 options:
1. The program can exit and you can run zypper from the command line.
2. The program can Run_zypper_After_Download Y/N, if this is Y zypper is run after the download **interactively**.
3. The program can Run_zypper_After_Download & Run_zypper_with_No_Confirm Y/N, if these are both Y zypper is run after the download **non-interactively**.
If zypper is ran by the program it has the advantage of capturing the output and sending it to the log file.
The program then exits with, the location and name of the logfile and “Normal zkk End”.

Errata Errata Errata Errata Errata Errata Errata

Q: Why is it called zkk?
A: I was going to call it ykk, but I was worried about trademarks from a zypper company ;).

Q: Why the dinosaur language Regina REXX, we want something cool like Rust or Ruby?
A: I had old code around which supported parallel execution and downloads and I knew how to do IPC with REXX.

Q: Why does zkk have a Thread and Child and the Child spawns separate stand alone programs (ZK_downLoader)?
A: I tried having zkk fork the ZK_downLoader program each one left a <defunct>, so you have the same number of <defunct> as you have patches. It looks really messy and what happens when you have thousands of patches? I do not know. The <defunct> do go away when parent zkk exits.

I bought a new fast router at the which leans to the upper part of the alphabet, it seems to speed up downloads and probably because the DNS is much faster, the timed out repositories seems to have stopped also, or maybe Suse improved their networking on the back end. The DNS timeout still happen on the router the Internet company provided me.

Repository files should be required to have enabled, name, baseurl entries. The snappy repo file does not have a name, we really need to save those 12 bytes (11 characters, an equal sign and a carriage return) ?!

Sometimes OpenSuse zypper prompts for the install ISO why? Can't you just pull the files down? You could obviously pull down the ISO file again.

zkk may or may not be faster then zypper, however it does provide an algorithm to rewrite it in a faster language, maybe with some improvements, zkk also provides the concept of multiple repositories with tracking the average download speed across each as a feedback loop. So instead of OpenSuse providing one baseurl they (OpenSuse) could create repository files with all the approved repositories in them and the end user running zypper could self discover the fastest sites for their location. I do not know how much it matters when the downloads come in parallel.

I had program crashes using Regina-REXX 3.9.1, I did put in a request to have the current version added to the zypper repos. That is why Regina-REXX 3.9.6 is checked for.

If Regina-REXX 3.9.1 is installed on the system I had to zypper remove regina-rexx and libregina3, I also remember rm /bin/regina, and then running which regina. I could run the *rpm --install libregina3-3.9.6-240341.1-x86_64-RHEL-7.rpm*
however I had to download the regina-rexx 3.9.6 tar ball expand it and *configure, make, make install.*
To get around the system reinstalling 3.9.1 again even when doing the *rpm --install regina-rexx-3.9.6-240341.1-x86_64-RHEL-7.rpm*
why, who knows.

For Regina REXX howto do a debug stream with multiple threads:

```
trace(?R)      /* Pause the program with a trace statement the ? does it */
Raw_Line = '54'
trace(O)       /* Turn tracing off */
Raw_Line = '54'
trace R         /* Turn tracing on it will just stream you do not need to push enter for each statement */
Raw_Line = '54'
```

Option 2

```
trace(?I)      /* Pause the program with a trace statement the ? does it */
Raw_Line = '54'
trace(O)       /* Turn tracing off */
Raw_Line = '54'
trace I         /* Turn tracing on it will just stream you do not need to push enter for each statement */
Raw_Line = '54'
```

Links to Opensuse Mirror List

Opensuse Official Repositories:
https://en.opensuse.org/Package_repositories

Opensuse Mirrors:
https://en.opensuse.org/openSUSE:Mirrors

OpenSuse Mirror Report:
https://mirrors.opensuse.org/