Random Access Concatenated Libraries and \textit{dd} Enable a Short-Latency, Content-Rich Website on an Inexpensive Shared Server

Don N. Krieger\textsuperscript{1}, University of Pittsburgh

\textit{RacLib} databases enable rapid record fetches with equivalent latency regardless of record location within a single file containing concatenated objects. A fast, reliable website housed on an inexpensive shared web server is described which provides short-latency access to more than 3,500,000 page images and a genealogical database containing more than 86,000,000 records.

Efficient means for website storage and subsequent delivery of graphic images is problematic when the image count is high. This is so because content-rich websites typically house their images either as individual files or as more costly binary database objects, both of which place high demands on storage resources with commensurate high monetary cost. These costs are reflected in file count limits imposed by hosting providers for inexpensive shared accounts, and storage space limits for more costly virtual private and dedicated server accounts.\textsuperscript{1,2} Cloud providers use this latter cost per unit storage space model as well.\textsuperscript{3,4,5} While shared accounts are offered for under US$10/month by many providers, websites which host more than 200,000 images almost always require resources...
only provided for accounts costing hundreds of U.S. dollars/month at minimum.

The work reported here using random access concatenated libraries (RacLibs) for image storage originated as a solution to limitations imposed by a shared web hosting service on a website designed to provide rapid access to more than 8,000 genealogical reference volumes with total page count in excess of 3.5 million. At current prices, hosting for the site using standard image storage methods would be at least US$1,000/month. At launch in 2010 the cost would have been much higher. The shared server solution uses standard common gateway interface (CGI)/shell-callable tools provided with the Unix and Linux operating systems.

RacLib files as large as 13 GB, each containing thousands of jpeg images, are handled as random-access memory using utility dd. Accessing individual images within large files using standard serial archive tools, for example, tar, is prohibitively slow.

dd was written by Ken Thompson and Dennis Ritchie as part of Bell Labs’ original Unix operating system version 5. It was propagated as one of the core utilities to all later versions of Unix and has been included in Linux since its inception. dd uses open(), lseek(), and read() to access files. The time required to access any data block within a large file using lseek() is independent of the location of the block within the file. Furthermore, the readout of a continuous set of records is handled as a streaming operation at the maximum speed provided by the file system. Hence, dd enables high speed utilization of large files as random-access memory.

The general usefulness of this capability is presented in solutions to 1) the page image problem mentioned previously, 2) a second, more complex, genealogical database search problem solved also in a shared web hosting environment, and 3) a neuroimaging research problem solved in a shared supercomputing environment.

A RacLib is a single file containing a collection of equal-length records. A record set is a contiguous sequence of one or more records. A second index file contains a one-line entry for each record set. At minimum, this line contains a label for the corresponding record set, the starting record number, and the number of records in the record set.

Each index entry may contain additional record set information. In some cases, for example, the social security death index (SSDI) problem (see next), the index file has sufficient intrinsic structure to enable organizing it also as a RacLib in which case its length may be unlimited. If not, the index file must be searched in a serial fashion to find the entry for the required RacLib record. In that case, the length of the index is best limited to 100,000 lines or less.

The file structures were designed for easy and efficient access using CGI script languages. Access to the files uses standard Linux tools, for example, grep and dd, hence comparable performance may be expected with any scripting language.

**IMAGE DELIVERY**

DonsList.net delivers human readable page images as shown in Figure 1 from genealogical resources, for example, census listings, military rolls, yearbooks, city, and phone directories. The website includes more than 3,500,000 page images from more than 8,000 titles. Each jpeg image is about 250 kbytes. For delivery at the request of a web browser user, an image must be extracted from the RacLib where it is stored and placed in a website-accessible cache directory as a jpg file. Management of the cache is detailed below.

The website is limited by the shared hosting service to 50,000 files. In addition, the response time of the website is limited by the load placed on the shared server by hundreds of other websites housed on the same server. One may compare the display capabilities and response time of DonsList with that provided by the Internet Archive, a well-funded library which houses more than 35 million volumes totaling billions of pages and which utilizes dedicated storage and compute servers.
To manage the file count, all of the page images from one or more of the titles are concatenated into a single file, a RacLib. The length of each page image is padded with null characters to an integer number of 1,024-byte records. Note that jpeg images are compressed, hence no additional compression need be applied to the RacLib to minimize its size.

Each line in the corresponding index file contains the book title, the page number within it, the starting record number, and the number of records for the page image, for example, “TallyHo1965 0404 4050730 348.” This example is page 404 from the 1965 Florida State University Yearbook. The image begins at record number 4050730 and contains 348 1,024-byte records. TallyHo1965 is one of the series of 1901–2007 Florida College for Women/FSU yearbooks housed on the website (Figure 1, upper right frame), 72 in total, all placed in a single RacLib and indexed in a single index file. The total number of page images in this RacLib and hence index lines is 21,988.

Fetch time from the RacLib to the image cache for the image shown in Figure 1 was 41 ms. The index line number for this page image is 15,582, that is, 41 ms included the time required for a serial search through 75% of the entries in the index file. The sequence of operations performed by each CGI instance in response to a webpage request are shown in Figures 2 and 3 and detailed as follows:

- Parse the request and capture the requested book title and page number.
- If the requested page image is present in the cache directory, generate the requested webpage with link to the cached image, send, and then terminate the current CGI instance.
- If the requested page image is not present:
  - Fetch the record number and count for the requested page image from the corresponding index file. Note that this fetch requires a serial search of the index file using Linux utility grep. The search matches the book title and page number, that is, the lines in the index file are content addressable.
  - Fetch the corresponding page image from the corresponding RacLib and write it to the cache directory. The RacLib is utilized at this step as random access memory by Linux utility dd.
  - Generate the requested webpage with link to the cached image, send, and then terminate the current CGI instance.

A TOKEN IS USED TO MINIMIZE THE CHANCE OF A RACE CONDITION IN WHICH TWO OR MORE CGI INSTANCES ATTEMPT TO PERFORM THE SAME FILE SYSTEM MANIPULATIONS AT THE SAME TIME.

**FIGURE 1.** An example DonsList.net webpage is shown with image (lower frame) of page 404 from the 1965 Florida State University Yearbook. The shared web server required 41 ms to extract the displayed 230-kbyte page image from a file containing more than 21,000 images and send it. See the text for details.
dd’s use of lseek() enables extraction of records with equivalent latency independent of the location within the file. Here are spot checks on the times required to fetch 230 1,024-byte records from near the beginning and end of a 6-GB file. Note that the test was run on the later part of the file first.

36 ms: Time dd if=FLStateU.RacLib bs=1,024 count=230 skip=6069853
41 ms: Time dd if=FLStateU.RacLib bs=1,024 count=230 skip=9398.

The file system is used as content-addressable memory since 1) the book title is present in the name of the corresponding RacLib and index files and 2) the page image name and page number are present in the name of the corresponding page image in the cache directory. This expedient markedly reduced the time and effort required to develop the CGI scripts. As mentioned previously, for delivery at the request of a web browser user, an image must be extracted from the RacLib where it is stored and placed in a cache directory as a jpg file. Management of the cache is handled as follows (see Figure 3). The current cache directory with page image files persists for 2,000 s. The directory name is selected using Linux utilities: `date +%s | sed -e “s/...$/ /”`. `date +%s` computes the number of 1000-s intervals that have elapsed since 1 January 1970 and `sed -e “s/...$/”` removes the last three digits.

Each CGI instance computes the directory name and checks for its existence. If an instance fails to find the cache directory, it attempts to grab the token file. This is accomplished by renaming the token file using the CGI instance’s unique process ID in the new temporary name. Releasing the token is accomplished by renaming the file to the original name. A token is used to minimize the chance of a race condition in which two or more CGI instances

**FIGURE 2.** The sequence of operations performed by each CGI instance in response to a webpage request. See the text for more detail. The “Cache Operations” indicated in the figure are shown in Figure 3.

**FIGURE 3.** The sequence of cache management operations performed by each CGI instance in response to a webpage request. This figure shows the “Cache Operations” indicated near the top of Figure 2. See the text for more detail.
After a system instance attempts to perform the same file system manipulations at the same time. The instance which grabs the token creates the current cache directory, deletes any cache directories which are more than 2,000 s old, releases the token, and continues. The CGI instances which fail to grab the token sleep for one second and check for the current directory again. Any CGI instance which fails a second time creates the token and grabs it. This ensures that the system will not hang indefinitely if a CGI instance which owns the token fails to release it. Note that this approach obviates the need for a daemon process which handles cache housekeeping.

This system has been in place for more than 12 years without a failure. During that period, it has delivered more than 11,000,000 page images, more than 9,800,000 fetched from a RacLib and 1,400,000 previously cached images. Figure 4 shows histograms of the response time of the system for both RacLib fetches (upper panel) and cache hits (lower panel). The cache hit mean response time is 24 ms faster than the RacLib fetch mean response time. The RacLib fetch response times from 2012–2013 are about 15 ms faster than those from 2020–2022, perhaps due to heavier load on the shared server and/or changes in the server’s disk subsystems over the years.

**SOCIAL SECURITY DEATH INDEX**

DonsList.net also houses a copy of the social security death index (SSDI). This is a database containing 86,806,853 64-byte records, that is, 5.5 GB. This is large enough to significantly degrade the performance of a relational database management system. The SSDI lists every deceased person with a U.S. social security number for whom the social security death benefit has been filed since the inception of the system in 1936. This version is current to 2011. Each decedent is listed with given name and surname, social security number, birth date, and death date. Figure 5 shows a sample search with results from DonsList. Note that the birth date and death month and year of the first entry in the results match those of U.S. Attorney General Robert F. Kennedy, and that the search required 39 ms. One may compare the search capabilities and response time of DonsList with those

---

**FIGURE 4.** The page image RacLib system has been in place since 2012 and has delivered more than 11,000,000 page images to date: more than 9,800,000 fetched from a RacLib and 1,400,000 previously cached images. Histograms of the response time of the system for both (a) RacLib fetches and (b) cache hits are shown for the first (blue) and last (red) 1,000,000 RacLib fetches and for the first and last 100,000 cache hits. Log-Log axes are used to enable appreciation of the relatively rare instances with prolonged response times. See the text for more details.
provided by large commercial libraries,\textsuperscript{15,16,17} all of which utilize dedicated storage and compute servers.

The SSDI data set possesses an intrinsic structure which is utilized to organize the RacLib and which also enables organizing the index file as a RacLib. The records in the data set are concatenated in the RacLib using a sort on the first letter of the surname, then the second letter of the surname, and then the first letter of the given name. All of the records with the same three key letters therefore fall together into one of $26^3$ record groups. Each index line contains the starting record number and the number of records in one of these 17,576 groups. The location of each index entry is calculated using the numbers corresponding to the three key letters, that is, the index is also a RacLib and randomly rather than serially accessed. For the search shown in Figure 5 on Robert Kennedy, the key letters are K(10), then E(4), then R(17). The number is the enumeration for each letter with A, the first letter of the alphabet, being 0 (zero). Using these numbers, the location of the corresponding index record is \(17 \cdot (26^2) + 4 \cdot (26) + 10 \cdot (26^2) = 6881\).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5.png}
\caption{An example SSDI search (lower frame) with results (upper right frame) is shown for “Robert Kennedy,” death years ranging from 1936 to 1974, and birth year 1925. The shared web server required 39 ms to extract the results and send them. See the text for details.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure6.png}
\caption{Sequence of operations performed by each CGI instance in response to an SSDI request. See text for more details.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure7.png}
\caption{The SSDI RacLib system has been in place since 2012 and has delivered more than 100,000 searches to date. A histogram of the response time of the system is shown. Log-Log axes are used to enable appreciation of the relatively rare instances with prolonged response times. See the text for more details.}
\end{figure}
Here is the sequence of operations performed by a CGI instance in response to an SSDI search request (Figure 6):

› Parse the request and capture the names and any other information provided for the search (see Figure 5).
› Compute the record number within the index RacLib using the three key letters and fetch the index record. This is a random access fetch using Linux utility `dd`.
› Use the record number and count contained in the index record to fetch the set of SSDI records from the SSDI RacLib using Linux utility `dd`.

1. Identify the xyz Coordinates Within a Specified Region
   Fetch all Index Lines Which Contain the Name of the Region
   Fetch all xyz Coordinates in the Corresponding cc Volumes

2. Identify the Region(s) Which Contain a Specific xyz Coordinate
   Parse the xyz Coordinates to Identify the Corresponding cc Volume
   Fetch all Lines Which Contain the Name of the cc Volume
   For Each cc Volume, Fetch all xyz coordinates and Filter for the specified xyz coordinates

**FIGURE 8.** The sequence of operations performed for two typical MRI RacLib searches. See the text for more details: 1) Identify the set of xyz coordinates within a specified region and 2) Identify the region(s) which contain a particular mm$^3$ volume specified with xyz coordinates.

**FIGURE 9.** Pairs of brain regions in this individual which are significantly coupled ($p < 10^{-8}$) are connected by chords. Green/red/blue chords connect cortical/subcortical/deep white matter regions. Coupling strength was measured by counting simultaneous appearance of electrical currents extracted from MEG recordings. Relative coupling strength is proportional to the circumferential length of the bars on the circle just inside the region names. (a) Results from 12 min of resting with eyes open in dim light. (b) Results from 12 min of cognitive task performance. See the text for more details.
Search the SSDI record set using all information provided for the search. Note that this is a serial search using Linux utility grep of all the SSDI records which match the three key letters.

Generate the requested webpage, send, and then terminate the current CGI instance.

One hundred twenty-nine of the SSDI record sets corresponding to a single three-letter index contain more than 100,000 64-byte records, that is, more than 6,400,000 bytes each. For example, the set of records for key letters "JO J" contains 142,989 records since it includes given names James, John, and Joseph and surnames Johnson and Jones. Over the decade that the SSDI RacLib system has been in place, that search (n = 148) required a mean of 149 ms with standard deviation of 150 ms. Figure 7 shows a histogram of the response time of the system. The searches which required more than 150 ms were likely primarily due to searches of these large record sets.

NEUROIMAGING

Anatomically stereotypic brain regions are automatically identifiable with millimeter accuracy using high-resolution magnetic resonance imaging (MRI) and computerized brain segmentation tools, for example, freesurfer and tracula. With these tools, the stereotypic region which contains each cubic millimeter of brain tissue is identified along with the xyz coordinates of the tissue in the MRI coordinate system. The xyz coordinate list for each region is typically stored in a separate file. For work carried out in our lab, the lists by regions were compiled into a single RacLib for each of more than 800 human subjects.

The xyz coordinates for each 1 mm³ volume are represented by a single string, for example, n41p12n35 represents (–41, 12, –35). The individual brain is divided into cubic centimeters, each containing up to 1,000 mm³ xyz coordinates, for example, n4_xp1_yn3_z represents (–4[0–9],1[0–9],–3[0–9]). Within the RacLib, all of the cubic centimeter volumes which overlap a single region are grouped together; all of the cubic millimeter xyz coordinates within a cubic centimeter volume are grouped together. Each entry in the index file contains the name of a region, the name of a cubic centimeter volume which overlaps the region, the starting record number, and the number of cubic millimeter xyz coordinates within the cubic centimeter volume. Here is an example index entry:

L_ctx_middletemporal n 4_xp1_yn3_z 85652 181.

The structure of the RacLib and its index enable easy and rapid searches of many types (Figure 8). The organization of the RacLib into record sets for each cubic centimeter volume coupled with the RacLib’s random access capability are particularly effective in reducing the load required to identify the region which contains a specific xyz coordinate, that is, (2) in Figure 8. Each record set contains a maximum of 1000 records which must be searched whereas the number of 1-mm³ volumes within a region ranges up to 60,000.

Use of RacLibs containing millimeter-resolution neuroimaging data as described was integral to developing the results shown in Figure 9. The figure shows pairs of brain regions in an individual (human) which are functionally coupled to each other during rest (upper panel) and cognitive task performance (lower panel). The functional measure is consistent occurrence of transient neuroelectric currents in two regions at once, p < 10⁻⁸ for each pair of regions. The currents were extracted from magnetoencephalographic (MEG) recordings. Note that fmajor is the only region which is connected from one side to the other during rest. fmajor is the posterior portion of the corpus callosum, a deep white matter tract made up of fibers which connect the two sides of the brain. In contrast, florid left/right regional coupling is seen during task performance. Note also that both left and right motor areas demonstrate coupling to other regions during task performance, for example, precentral, and cerebellum, but not during rest.

ABOUT THE AUTHOR

DON N. KRIEGER is a research scientist in the Department of Neurological Surgery, University of Pittsburgh, Pittsburgh, PA 15261 USA. His research interests are in advancing the understanding, diagnosis, and treatment of traumatic brain injury and includes extensive work in high-throughput supercomputing. Krieger received a Ph.D. in neuroscience from Maharishi International University. He serves as member and user representative of the Open Science Grid Council. Contact him at kriegerd@upmc.edu.

MARCH 2023 99
ACKNOWLEDGMENT
This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the Institutional Review Board of the University of Pittsburgh under Application No. PRO13070121.

REFERENCES
1. “Limitations on shared web hosting by different providers.” Mayo Host. Accessed: Jun. 2022. [Online]. Available: https://mayohost.com/blog/limitations-on-shared-web-hosting-by-different-providers/
2. “Account limits summary.” HostGator Support. Accessed: Jun. 2022. [Online]. Available: https://www.hostgator.com/help/article/account-limits-summary
3. “Azure storage blobs pricing.” Microsoft Azure. Accessed: Jun. 2022. [Online]. Available: https://azure.microsoft.com/en-in/pricing/details/storage/blobs/
4. “Amazon S3 simple storage service pricing.” Amazon Web Services. Accessed: Jun. 2022. [Online]. Available: https://aws.amazon.com/s3/pricing/
5. “Oracle cloud storage pricing.” Oracle. Accessed: Jun. 2022. [Online]. Available: https://www.oracle.com/in/cloud/storage/pricing/
6. M. S. Mahoney. “In the beginning: Unix at bell labs.” Accessed: Apr. 15, 2022. [Online]. Available: https://www.princeton.edu/~hos/Mahoney/unixpeople.htm
7. K. Thompson and D. Ritchie, Unix Programmer’s Manual, 5th ed., Bell Telephone Lab., Holmdel, NJ, USA, Jun. 1974, p. 51. Accessed: Apr. 15, 2022. [Online]. Available: https://www.computerhistory.org/collections/catalog/102641833
8. “GNU core utilities, src/dd.c.” GitHub. Accessed: Apr. 15, 2022. [Online]. Available: https://github.com/coreutils/coreutils/blob/master/src/dd.c
9. “1965 Florida State University Yearbook.” DonsList.net. Accessed: Jun. 2022. [Online]. Available: http://www.donslist.net/PGHLookups/cgi-bin/HandOff-3_0.cgi?FLStateU+TallyHo1965+0404+100W
10. “1965 Florida State University Yearbook.” The Internet Archive. Accessed: Jun. 2022. [Online]. Available: https://archive.org/details/tallyho181965florida/page/400/mode/2up
11. “dd(1) – Linux manual page.” Man7. Accessed: Apr. 15, 2022. [Online]. Available: https://man7.org/linux/man-pages/man1/dd.1.html
12. “11.2 dd: Convert and copy a file.” Manpagez. Accessed: Apr. 15, 2022. [Online]. Available: http://www.manpagez.com/info/coreutils/coreutils_74.php
13. M. Winand. “Performance impacts of data volume.” Luke. Accessed: Jun. 2022. [Online]. Available: https://use-the-index-luke.com/sql/testing-scalability/data-volume
14. “DonsList SSDI search page.” Accessed: Jun. 2022. [Online]. Available: http://www.donslist.net/PGHLookups/SSDIM.htm
15. “GenealogyBank SSDI search page.” [Online]. Available: https://www.genealogybank.com/explore/ssdi/all
16. “Fold3 SSDI search page.” Accessed: Jun. 2022. [Online]. Available: https://go.fold3.com/ssdi
17. “MyHeritage SSDI search page.” Accessed: Jun. 2022. [Online]. Available: https://www.myheritage.com/research/collection-10002/us-social-security-death-index-ssdi
18. B. Fischl, M. Sereno, and A. Dale, “Cortical surface-based analysis. II: inflation, flattening, and a surface-based coordinate system,” NeuroImage, vol. 9, no. 2, pp. 195–207, 1999, doi: 10.1006/nimg.1998.0396.
19. A. Yendiki et al., “Automated probabilistic reconstruction of white-matter pathways in health and disease using an atlas of the underlying anatomy,” Front. Neuroinformatics, vol. 5, p. 23, Oct. 2011, doi: 10.3389/fninf.2011.00023.
20. D. Krieger et al., “MEG-derived symptom-sensitive biomarkers with long-term test-retest reliability,” Diagnostics, vol. 12, no. 1, p. 84, 2022, doi: 10.3390/diagnostics12010084.