This document serves solely as a demonstration of AlphaClaim's technology and does not advocate legal or administrative action against any entity, including entities mentioned herein. This document neither alleges nor implies that any service or product from these entities infringes, either directly or indirectly, upon any patent or other intellectual property rights, including patents mentioned herein. This document neither alleges nor implies the invalidity or unenforceability of any patent, including patents mentioned herein.

This document was created independently by PriceWire, Inc. using AlphaClaim, for illustrative purposes only, without any third-party compensation or under any client contract or direction.

PriceWire, Inc. makes no representations or warranties regarding the completeness or accuracy of the information contained in this document and expressly disclaims any liability related to the use of such information for any purpose. PriceWire, Inc. is not a law firm, and this document does not provide legal advice.

US8332844B1 Infringement Case Study Report

This case study reviews how AlphaClaim was used to automatically claim chart 1,081 references in 45 minutes to find likely candidates for infringement of claim 7 of US8332844B1.

What is AlphaClaim?

AlphaClaim is a "brain" that computes accurate and detailed claim charts for a variety of document types against a given set of claims, adhering to a preponderance of the evidence standard. AlphaClaim has been aligned on 1000s of PTAB IPR institution and final written decisions.

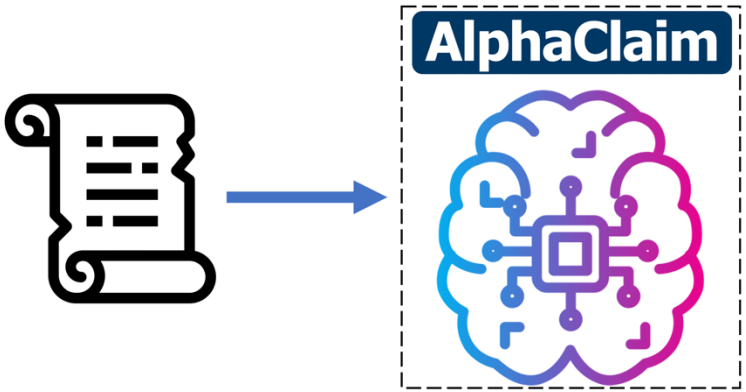
AlphaClaim performs automated, accurate, exhaustive claim charting of **superhuman quantities** of documents (often >10,000). For every document, AlphaClaim produces a claim chart with excerpts and explanations. It then computes an "AlphaScore" out of 5 for each claim element, indicating the strength of the document's disclosure of that element.

AlphaClaim can be applied in three ways. This document focuses on the third.

(1) To **identify the best invalidity grounds** (including combinations), for IPRs.
An AlphaScore of 4 or 5 indicates evidence stronger than the median IPR petition.

(2) To **identify evidence of patent validity** prior to assertion or sale of a patent.
If AlphaClaim's exhaustive review turns up low AlphaScores for all documents, a patent owner can be more confident that the patent will survive IPR or other 102/103 challenges.

(3) To **identify the best evidence of potential infringement**, for patent owners.
An AlphaScore of 4 or 5 indicates evidence stronger than the median filed claim chart.



Claim Element	Excerpt(s)	Alpha Score	AlphaClaim's Explanation
A method ...	"We describe an approach... "	5.0	The excerpt clearly meets the requirements...

How does AlphaClaim work?

AlphaClaim leverages many state-of-the-art AI technologies that are used in systems that achieve quality equivalent to the best humans, as detailed in the chart below. While computationally more expensive than consumer-grade chatbots, AlphaClaim achieves high quality with zero hallucination.

	ChatGPT-4	AlphaCode 2	AlphaGeometry	AlphaClaim
Company	OpenAI	Google	Google	AlphaClaim
Purpose	Chatbot	Coding contests	Math Olympiad	Claim charting
Quality Achieved	Mixed	Top 15% of ranked human competitors	Top ~60 mathematicians, worldwide	Skilled IP attorney
Specialization(s)	None	Fine-tuned, sampling-based	Custom pre-train, symbolic engine	Per-element score, sampling, context extraction
Hallucination Rate	Substantial	Zero	Zero	Zero
Computational Complexity	1x	>1,000x	>1,000x	>1,000x

AlphaClaim has 8 patents pending for its technological innovations.

Patent Overview & Infringement Evidence Identified

In this report, we show how in 45 minutes, AlphaClaim was able to review 1,081 references, ultimately producing two claim charts indicating potential infringement of claim 7 of the ‘844 patent.

Patent Overview (US8332844B2)

Originally issued to Panta Systems with a priority date of 12/30/2004. Currently owned by Intellectual Ventures (IV). Has been asserted against at least JP Morgan, Comerica, Liberty Mutual (in progress). IV’s argument for infringement of claim 7, according to their claim chart, appears to be that the claim is infringed by usage of the Docker container framework, a widely-deployed piece of open source software with several million users. No PTAB cases have been filed against the ‘844 patent yet.

How AlphaClaim was used

We used AlphaClaim to automatically, accurately, exhaustively claim chart 1,081 documents to look for potential infringement. After providing AlphaClaim a claim construction in technical language, which took 15 minutes, the AlphaClaim process took about 45 minutes. AlphaClaim charted all the references we provided it and by ranking all charted references based on their AlphaScores, AlphaClaim was able to find two documents which suggest potential infringement by each of VMware and Atlassian.

The evidence identified

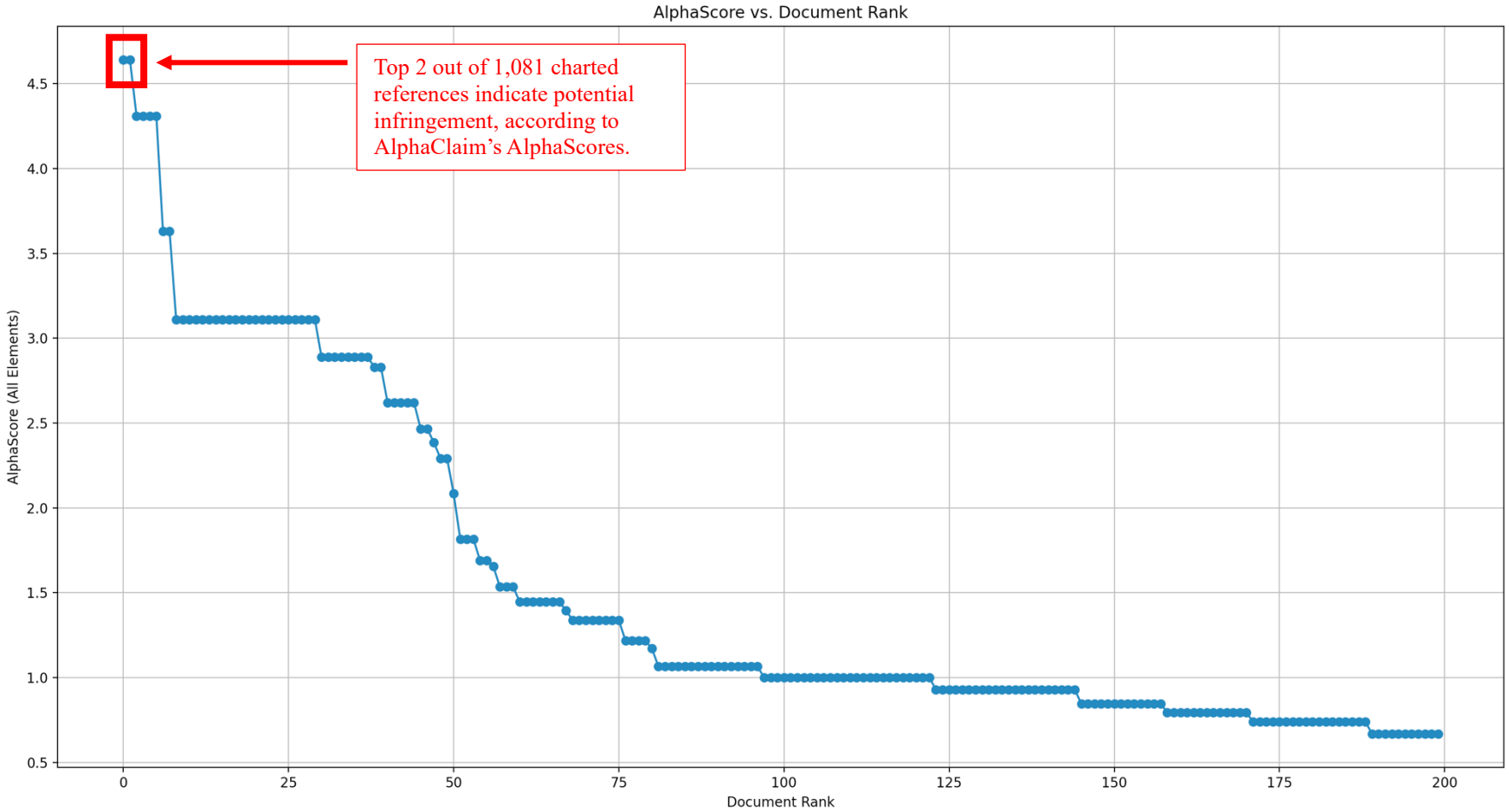
AlphaClaim found one patent from VMware and one application from Atlassian indicating that products from those companies may practice all elements of claim 7 of the ‘844 patent. None of these patents cited the ‘844 patent.

- For VMware, AlphaClaim identified US10574524B2 as disclosing a system which may potentially infringe claim 7. It is possible that the system disclosed in this patent corresponds to an implementation of the VMFS potentially using VMware Virtual SAN storage. While the patent itself does not disclose the final cache element of claim 7, VMware’s Virtual SAN product, suggested as one implementation of “storage 150” in the patent, includes a block read cache.
- For Atlassian, AlphaClaim identified US20200409686A1 as disclosing a system which may potentially infringe claim 7. It is possible that the system disclosed in this patent corresponds to Atlassian’s Bamboo CI/CD product, which continuously builds code using multiple VMs hosted on a “build system 106” and which are managed using delta disks. While the patent itself does not disclose the final cache element of claim 7, it is likely that the server on which the build system is deployed includes a disk block cache, as most any operating system such as Linux would. **It is noteworthy that AlphaClaim was able to identify this out-of-domain instance of potential infringement in a system buried within Atlassian’s product.**

To provide more insight into how AlphaClaim works, we show a score graph for the top 200 documents AlphaClaim charted. We then provide the unedited claim charts, AlphaScores, and explanations generated by AlphaClaim for the VMware and Atlassian documents.

Reviewed Documents Scores

AlphaClaim works by automatically, accurately, and exhaustively claim charting documents to find evidence of potential infringement. The graph shows the results of individual per-document AlphaScores for the documents reviewed against claim 7. While AlphaClaim reviewed 1,081 references for this case, we terminate the chart at around 200 documents for viewability, as scores drop too low below that. The per-reference AlphaScore is computed as a mean of the per-element AlphaScores within each reference.



AlphaClaim: Generated VMware Claim Chart

This is the unedited output from AlphaClaim. Any legal document would be drafted by a professional based on this output.

Note: the "claim construction" referenced by AlphaClaim is user-provided. This took about 15 minutes to create.

Claim Element (US8332844B1 claim 7)	AlphaClaim's Extracted Excerpt from US10574524B2	Alpha Score	AlphaClaim's Explanation
A method for providing data to a plurality of compute nodes, comprising:	In some embodiments, all VM images (i.e., base and delta VM images etc.) created are stored in persistent storage 197 such that any user with access to server 190 is able to use the Images to deploy VMs including operating systems and/or software applications that the user desires. In some embodiments, the VM images may be stored in storage 150 for use by users with access to host machine 110. Accordingly, when a user decides to deploy a VM with a certain operating system and/or software application, the user is able to conduct a search (using the metadata in the VM images) for the certain OS and software application using the user interface of image manager 195. In response to such a search by the user, image manager 195 may search through delta VM images in storage 197 or 150 to find one or more VM images that match the user's search.	5.0	The excerpt meets the claim construction's requirements by stating: - A provision mechanism is described as persistent storage (197, 150) and facilitated through a server (190) and a host machine (110), which aligns with the claim construction's requirement for a server, repository, or device that provides images. - The availability of these images for use by multiple machines is implied through the mention that any user with access to the server or host machine can use the images to deploy VMs. This indicates a shared or networked resource accessible by different users or systems, fulfilling the claim construction's requirement for images to be used by multiple machines.
storing blocks of a root image of said compute nodes on a first storage unit; storing leaf images for respective compute nodes on respective second storage units	FIG. 6 illustrates an example of a conceptual hierarchy of VM images including a base VM image 631 as well as a number of delta VM images created using the base VM image after updating the configuration of a VM created from the base VM image by the image manager, as described above. Base VM image 631 corresponds to base VM image 231 of FIG. 2, which is 15 GB in size and, as described above, may only contain code for running an operating system (e.g., Base OS). Similar to FIG. 2, each version X of the APP may be referenced to as App vX, Delta VM image 632 corresponds to an image created as a result of an update to the configuration of the VM (e.g., installing APP v1) instantiated using base VM image 631, as described above in relation to step 340 of FIG. 3. As shown delta VM image 632 is significantly smaller in size (e.g., 3 GB) than if an entire new VM image, such as VM image 232 of FIG. 2, had been captured from the complete image of the VM. Delta VM image 633 also corresponds to an image created as a result of additional updates (e.g., installing APP v2 after installing APP v1) to the state of the VM subsequent to the creation of delta VM image 632, as described in relation to FIG. 3. As	5.0	The Excerpt precisely matches the claim construction by describing: 1. A base VM image (base VM image 631) that acts as the root, base, or baseline image. This image contains data shared by many other images, fulfilling the first requirement of the claim construction. 2. Several delta VM images (delta VM images 632, 633, 634, and 635) that contain data unique to each respective machine, created after applying updates or changes to a VM instantiated from the base VM image. This fulfills the second requirement of the claim construction.

Claim Element (US8332844B1 claim 7)	AlphaClaim's Extracted Excerpt from US10574524B2	Alpha Score	AlphaClaim's Explanation
	shown, delta VM image 633 is also significantly smaller in size (e.g., 3 GB) than if an entirely new VM image, such as VM image 233 of FIG. 2, had been captured. As described above, in some case, instead of continuing to make further updates to the VM and creating additional delta VM images (e.g., delta VM image 633), the user may instantiate another VM from base VM image 631, update its configurations (e.g., install APP v2), and create delta VM image 634. Delta VM image 635 is another example of going back to instantiating a VM from base VM image 631, applying changes (e.g., installing APP v3), and creating a delta VM image.		
said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes;	At 320, image manager 195 may create a delta virtual disk file for the VM for storing writes resulting from future changes in the configuration of the VM. In some embodiments, once a delta virtual disk file is created, the parent virtual disk file(s) switches to a read-only mode and all future writes with respect to the VM are issued to the delta virtual disk file ... At 340, image manager 195 generates a delta VM image corresponding to the updated configuration of the VM. In some embodiments, the delta VM image (e.g., delta OVF package) comprises the delta virtual disk file as well as a metadata file (e.g., a dvf extension file) corresponding to the delta virtual disk file and the base VM image ... the metadata includes an identifier of the delta.VM image, where the identifier may be a short description about the delta VM image corresponding to the changes made by the user to the state of the VM (e.g., APP v1) ... In some embodiments, the metadata may also include a pointer to the delta virtual disk file, which itself may contain a pointer to the parent virtual disk file.	4.0	The Excerpt matches the claim construction in several key aspects: 1. It explains the creation of a delta virtual disk file specifically for storing changes (additional or modified data) to a VM, which aligns with the claim construction's requirement for explaining how delta images contain only additional or modified data. 2. It mentions that the parent virtual disk file becomes read-only after the creation of the delta virtual disk file, indicating that the delta file represents changes over the base configuration. 3. The generation of a delta VM image, which includes both the delta virtual disk file and a metadata file, is described. This process aligns with the claim construction's requirement for a satisfactory explanation of how delta images are generated or updated over time.
caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	Read Cache In a hybrid configuration, 70% of the cache device is dedicated to storing commonly used blocks. This reduces the I/O read latency incurred by slower disks. The goal of vSAN is to have a 90% cache hit rate. A cache hit is when a read request is found on the read cache.	N/A	This excerpt is from https://blogs.vmware.com/virtualblocks/2019/04/18/vsan-disk-groups/ (describing vSAN caching options). In this example, AlphaClaim was used to identify all elements except the cache limitation, because almost any system storing images on any disk would have some kind of cache present.

AlphaClaim: Generated Atlassian Claim Chart

This is the unedited output from AlphaClaim. Any legal document would be drafted by a professional based on this output.

Note: the "claim construction" referenced by AlphaClaim is user-provided. This took about 15 minutes to create.

Claim Element (US8332844B1 claim 7)	AlphaClaim's Extracted Excerpt from US20200409686A1	Alpha Score	AlphaClaim's Explanation
A method for providing data to a plurality of compute nodes, comprising:	In order to create the build agent in this manner, at step 302, the build manager 110 is configured to take a 'snapshot' of a build agent (either running or suspended) that is suitable for cloning ... At step 304, the base VM is converted into a template 202 to avoid accidental modifications and conserve resources ... Thereafter, at step 306, the template 202 can be cloned to create one or more linked clones 204 ... In addition to the attached virtual disks, a clone delta disk 210 is created for each linked clone 204 that is specific to the respective linked clone (at step 308) ... Finally, at step 310, the build manager 110 deploys the created linked clones on the host system 114 and initiates execution of the VMs.	4.0	The claim construction focuses on the provision of images for use by multiple machines. The excerpt describes a detailed process of creating and deploying virtual machine images (linked clones) from a template, which is derived from a base VM. This process inherently involves making these images (linked clones) available for use by multiple virtual machines on a host system. The build manager acts as the system component that manages this process, effectively serving as the "server" or "device" mentioned in the claim construction. The creation of clone delta disks for each linked clone allows for customization and storage of build-specific data, further aligning with the claim construction requirement of providing images for use by multiple entities. However, the excerpt does not explicitly mention the distribution or making available of disk, machine, OS, VM, or container images beyond the context of linked clones derived from a single template. The focus is more on the internal process of creating and deploying these clones rather than the provision of a variety of images to multiple machines.
storing blocks of a root image of said compute nodes on a first storage unit; storing leaf images for respective compute nodes on respective second storage units	In certain embodiments, the base template 202 includes an operating system virtual disk 206 that stores an image of the installed OS 120 and a software application virtual disk 208 that stores an image of the installed software applications 122. In other embodiments, the base template may include a single virtual disk that stores an image of the installed OS and the installed software. As depicted, each of the five linked clones 204 includes its own delta virtual disk 210A-210E (collectively referred to as clone delta disk 210) and each linked clone 204 is attached to the operating system virtual disk 206 and software application virtual disk 208 of the base template 202.	5.0	The Excerpt precisely matches the claim construction by describing a system that includes: 1. A base template (root, base, or baseline image) that contains data shared by many other images, specifically the operating system virtual disk and the software application virtual disk. 2. Delta virtual disks (delta, differential, incremental, or leaf images) that contain data unique to each respective linked clone (machine). The Excerpt explicitly states that each linked clone includes its own delta virtual disk, which stores data unique to that clone and is attached to the shared disks of the base template. This setup aligns with the claim construction's requirement for a system that stores

Claim Element (US8332844B1 claim 7)	AlphaClaim's Extracted Excerpt from US20200409686A1	Alpha Score	AlphaClaim's Explanation
			multiple images in a manner that includes a shared base image and unique delta images for each machine.
said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes;	A linked clone on the other hand is a copy of a virtual machine that shares virtual disks with the parent virtual machine in an ongoing manner. Accordingly, a linked clone is required to have continued access to the parent, without which the linked clone is disabled. This dependency or sharing of virtual disks conserves disk Space, and allows multiple virtual machines to use a single software installation. ... Each clone may have its own delta disk on which new or updated data can be written. In this way, multiple virtual machines can reuse the operating system and other files from their parent virtual machine's disk, but still perform its own operations upon creation independently of the base VM (and any other cloned VM).	4.0	The claim construction focuses on the concept of images (delta, differential, incremental, or leaf) that contain only additional or modified data relative to a baseline. The Excerpt, while not using these exact terms, closely aligns with this concept through the description of linked clones and delta disks. It explains that linked clones share a base with the parent VM and have delta disks for new or updated data, which matches the claim construction's requirement of containing only additional or modified data compared to a baseline.
caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	The *page cache* is the main disk cache used by the Linux kernel. In most cases, the kernel refers to the page cache when reading from or writing to disk. New pages are added to the page cache to satisfy User Mode processes's read requests. If the page is not already in the cache, a new entry is added to the cache and filled with the data read from the disk. If there is enough free memory, the page is kept in the cache for an indefinite period of time and can then be reused by other processes without accessing the disk.	N/A	This excerpt is from https://www.oreilly.com/library/view/understanding-the-linux/0596005652/ch15s01.html (describing Linux Page cache). In this example, AlphaClaim was used to identify all elements except the cache limitation, because almost any system storing images on any disk would have some kind of cache present.