

Compilación de documentos L^AT_EX en la nube

Diseño, aislamiento y evaluación de un servicio de compilación

Autor: Alex Rivera

Ciudad: Lima, Perú

Fecha: Abril de 2026

Resumen

Este informe describe el diseño y la evaluación de un servicio que compila proyectos $\text{\LaTeX}$ bajo demanda dentro de contenedores efímeros sin acceso a la red. Se presenta la arquitectura general, el modelo de aislamiento utilizado para ejecutar documentos de origen no confiable y una caché de artefactos direccionada por contenido. Las mediciones realizadas sobre un conjunto de 120 documentos muestran que, para documentos breves, el arranque del contenedor y no el motor de composición es el principal componente de la latencia, y que precalentar la imagen reduce la latencia mediana en un 38 %.

Índice general

Resumen	i
1. Introducción	1
1.1. Contexto	1
1.2. Objetivos	1
1.3. Alcance y estructura del informe	1
2. Diseño del servicio	3
2.1. Arquitectura general	3
2.2. Modelo de aislamiento	3
2.3. Clave de caché	4
2.4. Interfaz de programación	4
3. Evaluación y resultados	6
3.1. Metodología	6
3.2. Latencia	6
3.3. Comportamiento de la caché	7
3.4. Análisis de fallos	7
3.5. Conclusiones	7
Bibliografía	9

Índice de figuras

2.1. Arquitectura del servicio. El plano de compilación no tiene acceso a la red ni a los datos de otros usuarios.	3
3.1. Latencia mediana de compilación según el número de páginas.	7

Índice de tablas

2.1. Límites aplicados a cada compilación.	4
3.1. Latencia de extremo a extremo en segundos (mediana / percentil 95). .	6

Introducción

1.1 Contexto

Los sistemas de composición tipográfica basados en $\text{\TeX}$ siguen siendo la herramienta de referencia para publicar artículos científicos, tesis y documentación técnica (Lamport, 1994). Sin embargo, instalar y mantener una distribución completa de $\text{\TeX}$ Live exige varios gigabytes de disco y un conocimiento considerable de su ecosistema de paquetes. Para una organización que solo necesita transformar una plantilla y unos datos en un archivo PDF (por ejemplo, un sistema de generación de certificados o de informes periódicos), esa carga resulta desproporcionada.

Durante los últimos años han aparecido editores en línea que centralizan la compilación (Overleaf, 2023), pero sus componentes internos no se exponen como piezas reutilizables. Este informe estudia la compilación como un servicio independiente, sin estado, que recibe un conjunto de archivos y devuelve el PDF resultante junto con un diagnóstico estructurado de errores y advertencias.

1.2 Objetivos

El trabajo persigue tres objetivos concretos:

- O1.** Diseñar un modelo de ejecución que permita compilar documentos de origen no confiable en una infraestructura compartida sin comprometer el servidor ni los datos de otros usuarios.
- O2.** Garantizar que una misma entrada produzca siempre la misma salida mientras no cambie la versión de la imagen de compilación.
- O3.** Reducir la latencia percibida mediante una caché de artefactos que evite recompilar documentos que no han cambiado.

1.3 Alcance y estructura del informe

El alcance se limita a los tres motores más utilizados ($\text{pdf\TeX}$, $\text{Xe\TeX}$ y $\text{Lua\TeX}$) y al flujo de trabajo gestionado por `latexmk`. Quedan fuera del estudio las funcionalidades

que requieren ejecutar programas externos durante la compilación, como `minted` o la externalización de figuras de TikZ, porque el modelo de aislamiento las prohíbe de forma deliberada.

El resto del documento se organiza de la siguiente manera. El capítulo 2 describe la arquitectura del servicio, el modelo de aislamiento y la clave de caché. El capítulo 3 presenta la metodología de evaluación, los resultados obtenidos y las lecciones aprendidas durante seis meses de operación.

Diseño del servicio

2.1 Arquitectura general

El servicio se compone de tres planos claramente separados: el plano de entrada, que valida y normaliza los proyectos recibidos; el plano de compilación, formado por contenedores efímeros; y el plano de almacenamiento, donde se guardan los artefactos generados. La figura 2.1 muestra la relación entre ellos.

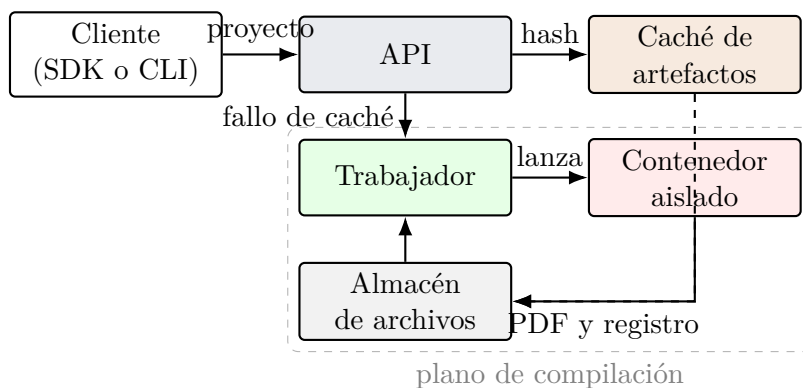


Figura 2.1: Arquitectura del servicio. El plano de compilación no tiene acceso a la red ni a los datos de otros usuarios.

Cada petición se procesa en cuatro pasos. Primero, la API comprueba los límites de tamaño y rechaza rutas peligrosas (referencias al directorio superior, rutas absolutas o enlaces simbólicos). Segundo, calcula un resumen criptográfico del proyecto normalizado. Tercero, consulta la caché y, si existe un artefacto para ese resumen, lo devuelve de inmediato. Cuarto, en caso contrario, un trabajador lanza un contenedor, ejecuta la compilación y almacena el resultado.

2.2 Modelo de aislamiento

Un documento $\text{\LaTeX}$ es, en la práctica, un programa: puede leer y escribir archivos, consumir memoria sin límite y, si el intérprete lo permite, ejecutar órdenes del sistema. Por ello cada compilación se ejecuta en un contenedor nuevo con las restricciones de la tabla 2.1.

Tabla 2.1: Límites aplicados a cada compilación.

Recurso	Mecanismo	Límite
Tiempo de pared	<code>timeout -s KILL</code>	60 s (síncrono) / 300 s (asíncrono)
Memoria	<code>cgroup v2</code>	2 GiB
Procesadores	<code>cgroup v2</code>	2 vCPU
Procesos	<code>pids.max</code>	64
Red	espacio de nombres	ninguna
Llamadas al sistema	<code>seccomp</code>	lista blanca
Ejecución de programas	<code>shell_escape = f</code>	deshabilitada

Además de los límites de recursos, el sistema de archivos raíz del contenedor es de solo lectura y el directorio de trabajo se monta en memoria. La opción `openout_any = p` del archivo `texmf.cnf` impide que el motor escriba fuera de ese directorio, y `-no-shell-escape` refuerza en la línea de órdenes la prohibición ya establecida en la configuración.

2.3 Clave de caché

Sea $B = \{(p_i, b_i)\}_{i=1}^n$ el conjunto de archivos del proyecto, donde p_i es la ruta relativa normalizada y b_i su contenido en bytes. Dado el motor e y la versión de la imagen v , la clave de caché se define como

$$\kappa(B, e, v) = H\left(e \parallel v \parallel \left\|_{i=1}^n \left(|p_i| \parallel p_i \parallel |b_i| \parallel b_i\right)\right\| \right), \quad (2.1)$$

donde H es SHA-256 y $\parallel$ denota la concatenación con prefijos de longitud. Incluir v en la clave garantiza que una actualización de T_EX Live invalide los artefactos antiguos, que son la principal fuente de resultados no reproducibles en proyectos de larga duración.

2.4 Interfaz de programación

El listado 2.1 muestra una petición mínima al servicio. El cuerpo contiene los archivos del proyecto codificados como texto y el nombre del archivo principal; la respuesta es el PDF o, si se solicita mediante la cabecera `Accept`, un objeto JSON con los diagnósticos.

Listado 2.1: Petición de compilación con curl.

```

1 curl -X POST https://latex-api.example/v1/compile \
2   -H "Authorization: Bearer $API_KEY" \
3   -H "Content-Type: application/json" \
4   -d '{
5       "engine": "pdflatex",

```

```
6      "entrypoint": "main.tex",
7      "files": [
8        { "path": "main.tex", "content": "\\documentclass{
          article}..." }
9      ]
10    }, \
11    -o salida.pdf
```

Cuando la compilación falla, el servicio devuelve un documento de problema conforme a RFC 9457 con el código `compile_failed` y la lista de diagnósticos, cada uno con su archivo, línea y mensaje. De este modo el cliente puede señalar el error directamente en el editor sin analizar el registro completo del motor.

Evaluación y resultados

3.1 Metodología

Se recopilieron 120 proyectos públicos de distinto tipo (artículos de congreso, artículos de revista, currículos y tesis) y se compiló cada uno diez veces en una máquina virtual de dos procesadores virtuales y 4 GiB de memoria. Las ejecuciones se dividieron en dos escenarios:

- **Frío:** contenedor nuevo, sin archivos de formato pregenerados ni caché de fuentes.
- **Caliente:** imagen con los formatos de los tres motores y la base de datos de fuentes de `luaotfload` generados durante la construcción de la imagen.

Para cada ejecución se registró el tiempo de cada fase: creación del contenedor, carga del formato, pasadas del motor, procesamiento de la bibliografía y escritura de los artefactos.

3.2 Latencia

La tabla 3.1 resume la latencia de extremo a extremo por tipo de documento. La creación del contenedor cuesta alrededor de 1,1 s independientemente del documento, y el motor solo domina el tiempo total en documentos de más de cuarenta páginas.

Tabla 3.1: Latencia de extremo a extremo en segundos (mediana / percentil 95).

Tipo de documento	Páginas	Frío	Caliente
Artículo de congreso sin bibliografía	4–6	5,8 / 7,9	2,9 / 3,6
Artículo de congreso con <code>BIBTEX</code>	6–8	9,7 / 12,4	6,1 / 7,8
Artículo de revista con <code>biber</code>	10–14	14,2 / 18,0	10,5 / 12,9
Currículo (<code>fontspec</code> , <code>LuaTeX</code>)	1–2	8,3 / 10,1	4,0 / 5,2
Tesis	80+	41,6 / 55,3	36,9 / 48,7

La figura 3.1 muestra la latencia mediana en función del número de páginas para ambos escenarios. La diferencia entre las curvas se mantiene aproximadamente constante, lo que confirma que las optimizaciones actúan sobre el coste fijo de arranque y no sobre el tiempo de composición.

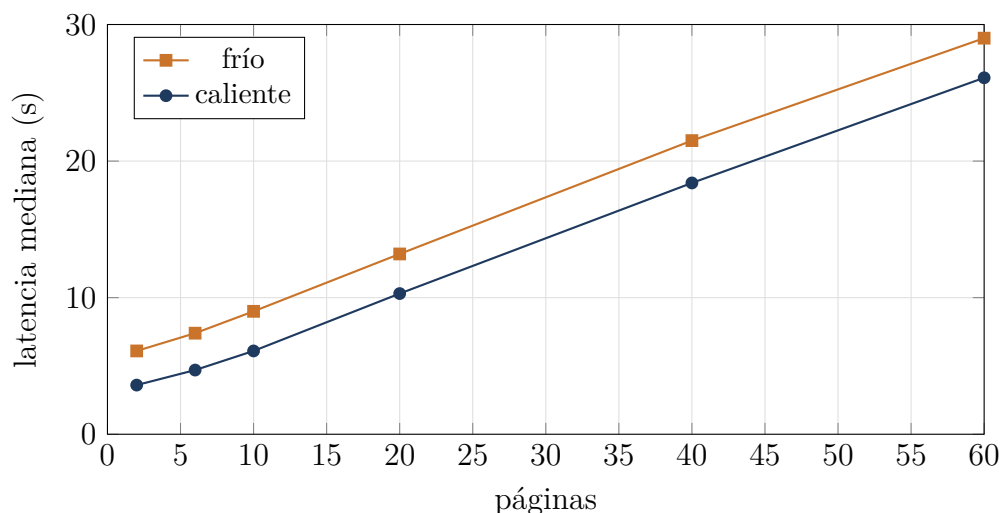


Figura 3.1: Latencia mediana de compilación según el número de páginas.

3.3 Comportamiento de la caché

Durante seis meses de operación, el 68 % de las peticiones se resolvieron desde la caché. La tasa de aciertos varió notablemente según el tipo de cliente: las plataformas docentes que regeneran el PDF de una tarea en cada visita alcanzaron el 91 %, mientras que los editores interactivos que compilan en cada pulsación se mantuvieron por debajo del 20 %. La inclusión de la versión de la imagen en la ecuación (2.1) provocó un vaciado completo de la caché en cada actualización de T_EX Live; la tasa de aciertos recuperó su valor habitual en unas 36 horas.

3.4 Análisis de fallos

De las compilaciones que no estaban en caché, el 11,4 % terminó con error. Los diagnósticos atribuyeron el 54 % de esos errores a archivos inexistentes referenciados con `\includegraphics` o `\input`, el 23 % a secuencias de control no definidas y el 9 % a paquetes que requieren la ejecución de programas externos. Solo el 0,3 % de las compilaciones fue interrumpido por el tiempo límite y ninguna superó los límites de memoria o de procesos una vez ajustados los valores de la tabla 2.1.

3.5 Conclusiones

Los resultados permiten extraer tres conclusiones principales:

1. En documentos breves, la sobrecarga de infraestructura pesa más que la composición tipográfica; precalentar la imagen elimina la mayor parte de ese coste.

2. Una caché direccionada por contenido resulta muy rentable porque las cargas de trabajo reales vuelven a solicitar documentos sin cambios con mucha más frecuencia de la que los modifican.
3. El aislamiento estricto no supuso un obstáculo en la práctica: menos del 10 % de los errores se debieron a funcionalidades prohibidas, y en todos los casos existía una alternativa compatible.

Como trabajo futuro se propone la compilación incremental por páginas y la transmisión de los diagnósticos mientras el motor sigue en ejecución, de forma que el usuario reciba el primer error antes de que termine la compilación completa (Knuth, 1986).

Bibliografía

Knuth, D. E. (1986). *The T_EXbook*. Addison-Wesley, Reading, MA, EE. UU.

Lamport, L. (1994). *ΛT_EX: A Document Preparation System*. Addison-Wesley, Reading, MA, EE. UU., 2 edition.

Overleaf (2023). CLSI: Common LaTeX service interface. <https://github.com/overleaf/overleaf>. Consultado el 15 de enero de 2026.