

# make.com Parallel Processing Strategie clever nutzen und beschleunigen

Category: Tools

geschrieben von Tobias Hager | 10. Dezember 2025



# make.com Parallel Processing Strategie clever nutzen und beschleunigen: Der

# ultimate Guide für Workflow-Power-User

Du willst deine make.com Workflows endlich aus der lahmen Warteschlange holen? Vergiss die netten Marketing-Versprechen und lies, wie du mit der richtigen Parallel Processing Strategie aus deinem Automatisierungs-Setup eine echte Performance-Maschine machst. Schluss mit Flaschenhälsen, Zeitfressern und ineffizientem Click-the-donkey – hier kommt der kompromisslose Deep Dive, der Make-User endlich auf das nächste Level katapultiert. Bereit für Automation ohne Kompromisse? Dann schalt den Turbo ein.

- Was Parallel Processing bei make.com wirklich ist – und warum du es garantiert unterschätzt
- Die wichtigsten technischen Voraussetzungen und Limitierungen von Parallelausführung in Make
- Typische Fehlerquellen und die größten Performance-Killer bei parallelen Workflows
- Step-by-Step: Wie du Parallel Processing in Make-Workflows implementierst und optimierst
- Best Practices für skalierbare, fehlerresistente Automatisierungen mit Parallelisierung
- Wie du mit cleverem Modul-Design und Queue-Strategien die Ausführungsgeschwindigkeit explodieren lässt
- Monitoring und Troubleshooting: So erkennst du Bottlenecks und Performance-Lecks frühzeitig
- Warum 80% aller Make-User Parallelisierung falsch nutzen – und wie du es besser machst
- Die wichtigsten Tools, Module und Settings für echte Parallel Power
- Fazit: Parallel Processing als Schlüssel für echte Automatisierungs-Effizienz

make.com hat sich in den letzten Jahren als das Schweizer Taschenmesser für No-Code-Automatisierung etabliert. Aber während die meisten Nutzer noch brav ihre linearen Szenarien abarbeiten lassen, verschenken sie das wahre Potenzial der Plattform: Parallel Processing. Klingt komplex? Ist es auch – zumindest, wenn du mehr willst als den x-ten “Wenn-Mail-dann-Trello”-Workaround. Wer 2024 noch mit sequentiellen Workflows hantiert, verliert Zeit, Geld und Nerven. Der Unterschied zwischen Workflow-Spielplatz und skalierbarer Automatisierungs-Architektur? Exakt das: Parallele Verarbeitung. In diesem Artikel zerlegen wir die Technik, zeigen die Fallstricke, räumen mit Mythen auf und liefern dir eine Anleitung, die garantiert nicht im offiziellen Doku-Zuckerguss zu finden ist. Willkommen im Maschinenraum von make.com – hier wird nicht geklickt, hier wird gehackt.

# Was ist Parallel Processing bei make.com? – Das unterschätzte Herzstück moderner Automatisierung

Parallel Processing, oder parallele Verarbeitung, beschreibt die Fähigkeit, mehrere Aufgaben gleichzeitig auszuführen. In der make.com-Welt bedeutet das: Module in einem Szenario werden nicht brav nacheinander abgearbeitet, sondern in mehreren "Execution Threads" simultan durchgeprügelt. Das klingt nach Raketenwissenschaft, ist aber der Gamechanger, wenn du große Datenmengen, langsame APIs oder Timeouts im Griff behalten willst. Der Hauptvorteil? Massive Zeitersparnis, bessere Hardware-Auslastung und spürbar schnellere Workflows.

Das make.com-Interface ist darauf ausgelegt, Prozesse zu visualisieren und logisch zu strukturieren. Was viele nicht wissen: Die Plattform entscheidet selbst, ob und wie Module parallel ausgeführt werden – abhängig von deren Typ, Konfiguration und der verwendeten Iterator-Logik. "Iterator", "Array Aggregator", "Router", "Error Handler" – das sind nicht nur hübsche Icons, sondern die zentralen Werkzeuge für echtes Parallel Processing. Besonders Iterator und Router sind die Gatekeeper: Sie splitten Datenströme und verteilen sie auf parallele Ausführungspfade.

Der Knackpunkt: make.com propagiert Parallelisierung gerne als Feature, verschweigt aber die technischen Limitierungen, die mitspielen. Jede Organisation, jedes Szenario und selbst jeder Account hat harte Limits, was gleichzeitige Ausführungen angeht (Stichwort: Concurrent Executions, Rate Limits, API-Quotas). Wer hier nicht sauber plant und testet, erlebt böse Überraschungen. Und nein, die Standard-Hilfeartikel kratzen nur an der Oberfläche. Wer richtig skalieren will, muss verstehen, wie make.com unter der Haube tickt.

Wichtig: "Parallel" heißt in der Praxis eben nicht "unendlich gleichzeitig". make.com kapselt die Ausführung in virtuelle Maschinen, die je nach Tarif, Szenario-Komplexität und Ressourcen unterschiedlich viel Power bekommen. Wer's übertreibt, landet schneller im "Scenario Execution Limit Reached"-Fehler als ihm lieb ist. Die Kunst liegt also darin, die Architektur so zu designen, dass parallele Prozesse effizient, aber kontrolliert ablaufen – ohne Deadlocks, Race Conditions oder Datenchaos.

## Technische Voraussetzungen und

# Limitierungen – Die bittere Wahrheit der make.com Parallelisierung

Bevor du jetzt voller Euphorie deinen ersten 500-fachen Parallel-Workflow startest: Stop! Parallele Verarbeitung in make.com ist kein Wunschkonzert. Die Plattform setzt dir technische Grenzen, die du kennen und respektieren musst, sonst schießt du dich schneller ins Aus als du "Module Limit" sagen kannst. Die wichtigsten Limitierungen für Parallel Processing bei make.com sind:

- Maximale gleichzeitige Szenario-Ausführungen pro Organisation (abhängig vom Tarif)
- Begrenzte Ressourcen pro Szenario (Memory, CPU, Execution Timeouts)
- API-Rate-Limits der angebundenen Dienste (Google, Microsoft & Co. sind da gnadenlos)
- Interne make.com Limits für Iterator- und Aggregator-Module (z. B. maximal 1000 Items pro Iterator-Ausführung)
- Globale Throttling-Regeln für besonders frequentierte Workflows (Drosselung bei Lastspitzen)

Du willst wissen, wie viele parallele Threads dein Account wirklich stemmen kann? Die Antwort: Das weiß nur make.com – und die rücken die echten Zahlen nicht raus. Erfahrungsgemäß gilt: Free- und Core-Pläne laufen mit angezogener Handbremse, Pro- und Enterprise-Accounts bekommen mehr Power, aber auch hier gibt es unsichtbare Deckel. Wer's auf die Spitze treibt, bekommt gern mal eine nette "Contact Support"-Mail, weil das eigene Szenario das Rechenzentrum in die Knie zwingt.

Ein weiteres Problem: API-Quotas und Rate-Limits der integrierten Tools. Wenn du z. B. 500 Mails gleichzeitig via Gmail-Modul verschickst, ist spätestens nach 100 Schluss – Google blockiert, und dein Szenario knallt in die Fehlerbehandlung. Das gleiche gilt für CRM-Updates, Datenbank-Queries oder Cloud-Speicher: Wer zu viel parallelisiert, wird gebremst oder sogar temporär ausgesperrt. Die Lösung? Intelligentes Queue-Management, Batch-Processing und der gezielte Einsatz von Delay- und Sleep-Modulen, um Requests zu kanalisieren.

Last but not least: Memory und Execution Timeouts. make.com killt jede Ausführung, die zu viel Speicher braucht oder zu lange läuft. Parallele Verarbeitung erhöht die Last auf das System exponentiell – und das ist der Punkt, an dem die meisten "Power User" auf die Nase fallen. Wer komplexe Workflows parallelisieren will, muss sie modularisieren und sauber kapseln. Sonst ist der Performance-Gewinn schneller weg als der Kaffee in einer Digitalagentur am Montagmorgen.

# Typische Fehlerquellen bei paralleler Verarbeitung – So killst du deine Performance garantiert

Die Theorie klingt einfach: "Ich mach's einfach parallel, dann wird's schneller." Willkommen in der Realität: 80% aller Make-User, die Parallel Processing nutzen, machen es falsch. Was läuft schief? Hier die größten Fehlerquellen, die garantiert jeden Versuch der Parallelisierung sabotieren:

- Unkontrollierte Nutzung von Iterator- und Router-Modulen ohne Limitierung der Durchsatzrate
- Fehlendes Error Handling – ein Fehler im Thread killt den gesamten Workflow
- Race Conditions: Mehrere parallele Prozesse schreiben auf dieselbe Ressource (z. B. Google Sheet, Datenbank), was zu Datenmüll und Konflikten führt
- Unzureichendes Monitoring: Niemand merkt, dass die Hälfte der Prozesse im Timeout versackt
- Ignorieren von API-Limits: Zu viele Requests in kurzer Zeit, geblockt vom Drittanbieter

Gerade bei Iterator-Modulen ist der Fehler fast schon vorprogrammiert. Viele denken: "Je mehr Items, desto schneller, wenn alles parallel läuft." Falsch. Iterators in make.com splitten Daten zwar auf, aber die tatsächliche Ausführung hängt von den Limits der Plattform und der Ziel-API ab. Wer 10.000 Datensätze in einen Google Sheet ballert, darf sich nicht wundern, wenn nach 300 Zeilen die API dichtmacht. Die richtige Strategie: Batching und Throttling – also Datensätze in Paketen verarbeiten und zwischen den Requests gezielt Pausen einbauen.

Ein weiteres Desaster: Fehlende Fehlerbehandlung. In parallelen Workflows ist Error Handling kein optionales Feature, sondern lebenswichtig. Ohne saubere Error Handler-Module läuft ein einziger Fehler Amok und killt im schlimmsten Fall den gesamten Prozess. Die beste Lösung: Fehler isolieren, Logging ergänzen, und Failed Items gezielt in eine Recovery-Schleife schicken.

Und noch ein Klassiker: Race Conditions. Wer mehrere parallele Prozesse auf die selbe Datenbank, das gleiche Sheet oder eine API schreibt, riskiert Datenkorruption, doppelte Einträge oder fehlerhafte Ergebnisse. Hier hilft nur: Ressourcen sperren (Locking), Datenflüsse entkoppeln oder – ganz Oldschool – mit dedizierten Queues für jeden Prozess arbeiten.

# Step-by-Step: So implementierst und optimierst du Parallel Processing in make.com

Genug Theorie – jetzt wird's praktisch. Du willst echtes Parallel Processing in make.com ausrollen? Hier die Schritt-für-Schritt-Anleitung, wie du deine Workflows clever parallelisierst, ohne dass alles explodiert:

- 1. Szenario-Architektur planen: Überlege, welche Module parallelisierbar sind (z. B. unabhängige Datenverarbeitung, API-Calls) und wie Datenflüsse voneinander entkoppelt werden können.
- 2. Iterator- und Router-Module sinnvoll einsetzen: Nutze Iteratoren, um Listen in Einzel-Items zu zerlegen, und Router, um verschiedene Verarbeitungspfade aufzuteilen.
- 3. Limits der Zielsysteme beachten: Prüfe API-Quotas und baue Delay- oder Sleep-Module ein, um die Request-Rate zu steuern.
- 4. Fehlerbehandlung integrieren: Jeder parallele Pfad braucht eigene Error Handler, Logging und ggf. eine Recovery-Funktion für fehlgeschlagene Datensätze.
- 5. Batch Processing nutzen: Große Datenmengen in kleine Pakete splitten (z. B. 50–100 Items pro Batch), damit keine Timeouts oder Memory-Overflows entstehen.
- 6. Monitoring und Alerts einrichten: Analysiere Laufzeiten, Fehler, Bottlenecks und setze automatische Benachrichtigungen für kritische Fehler.

Beispiel: Du willst 10.000 Datensätze aus einer Datenbank in verschiedene Systeme pushen. Die naive Lösung: Ein Iterator zerteilt alles, dann rennen 10.000 Threads los. Die schlaue Lösung: Erst mit Array Aggregator in 100er-Paketen batchen, dann mit Iterator parallelisieren, Delay/Retry einbauen, Error Handling pro Batch und separate Recovery Queue für Fehlerfälle.

Das Ergebnis: Keine API-Blocks, stabile Performance, kein Datenverlust. Und vor allem: Ein Workflow, der auch bei Lastspitzen kontrolliert und nachvollziehbar bleibt.

## Best Practices: Skalierbare und fehlerresistente Workflows

# mit Parallel Processing

Wer auf Dauer wirklich von Parallel Processing profitieren will, braucht mehr als ein paar schicke Icons im Szenario-Builder. Hier die wichtigsten Best Practices, die deine make.com Workflows auf Enterprise-Level heben – garantiert ohne Marketing-Blabla:

- Modulare Szenario-Architektur: Zerlege große Workflows in kleine, unabhängige Teil-Szenarien, die gezielt parallelisiert werden können. So vermeidest du Monster-Workflows mit unkontrollierbarer Fehlerkette.
- Queues und Puffer nutzen: Komplexe Datenflüsse über Message Queues oder dedizierte Buffer-Module steuern. Damit verhinderst du, dass einzelne Threads Ressourcen blockieren oder sich gegenseitig ausbremsen.
- Idempotente Operationen: Alle Schreibvorgänge so gestalten, dass Wiederholungen keine Fehler verursachen (z. B. Insert mit Unique Key, Upsert statt Update/Delete), um Race Conditions zu minimieren.
- Monitoring-Tools extern anbinden: Logs und Alerts an externe Systeme (z. B. Datadog, Grafana, Slack) senden, damit du Fehler, Timeouts und Bottlenecks in Echtzeit siehst.
- API-Quotas und Limits zentral verwalten: Zentrale Variable für Request-Limits und Throttle-Werte im Szenario nutzen, statt überall hart zu codieren. So passt du die Performance bei Bedarf an, ohne alles umzubauen.

Und der wichtigste Tipp: Testen, testen, testen! Parallele Workflows müssen regelmäßig unter Lastbedingungen geprüft werden. Simuliere Worst-Case-Szenarien, provoziere Fehler, analysiere die Logs – nur so findest du die echten Schwachstellen, bevor sie in Produktion knallen.

Fazit: Wer Parallel Processing clever nutzt, gewinnt nicht nur Geschwindigkeit, sondern auch Skalierbarkeit, Ausfallsicherheit und ein echtes Maß an Automatisierungs-Professionalität. Die meisten Probleme entstehen nicht durch zu wenig Parallelisierung, sondern durch schlecht geplante Parallelisierung. Wer die Technik versteht, macht aus make.com ein echtes Kraftwerk.

## Monitoring, Troubleshooting und Performance-Tuning – Kontrolliere deine parallelen Prozesse

Du hast die Parallelisierung im Griff – aber wie bleibt der Workflow auch nach Monaten stabil? Ohne Monitoring und Troubleshooting ist jeder noch so schnelle Workflow ein Pulverfass. make.com bietet zwar rudimentäre Logs, aber die reichen bei komplexen, parallelen Szenarien selten aus. Die besten

## Monitoring- und Troubleshooting-Strategien:

- Alle kritischen Punkte im Workflow mit Logging-Modulen versehen (z. B. Write to Google Sheet, HTTP Request mit Response-Logging)
- Eigene Monitoring-Szenarien bauen, die regelmäßig Statusdaten und Fehler auslesen und an zentrale Dashboards schicken
- Alerts für Timeouts, Fehlerquoten oder API-Blocks einrichten (Push-Nachricht, Mail, Slack)
- Performance regelmäßig messen: Laufzeiten, Fehlerhäufigkeit, Durchsatz pro Thread
- Fehlerhafte Items in separate Recovery-Queues schieben, um sie später erneut zu verarbeiten

Für echte Power-User empfiehlt es sich, die internen Logs mit externen Tools wie Datadog, Prometheus oder Grafana zu kombinieren. So siehst du auf einen Blick, wo Bottlenecks entstehen, welche Threads regelmäßig abstürzen und wo Optimierungspotenzial besteht.

Performance-Tuning ist keine Einmal-Aktion, sondern ein Prozess. Passe Batch-Größen, Delay-Zeiten und Re-Retry-Intervalle regelmäßig an die tatsächliche Systemlast an. Die Make-API-Statistiken helfen, Flaschenhälse zu identifizieren und gezielt zu beseitigen. Und vergiss nicht: Alles, was du nicht misst, kannst du nicht optimieren – Monitoring ist der Schlüssel zu nachhaltiger Automatisierung.

# Fazit: Parallel Processing als Schlüssel zur echten make.com Automatisierungs-Effizienz

Wer make.com wirklich ausreizen will, kommt an Parallel Processing nicht vorbei. Parallele Verarbeitung ist nicht nur ein Performance-Hack, sondern das Fundament für skalierbare, robuste und zukunftssichere Workflows. Die Technik dahinter ist anspruchsvoll – und Make macht es dir alles andere als leicht, die Limits und Fallstricke zu erkennen. Aber wer sich mit Iterator, Router, Error Handler, Queue- und Batch-Strategien ernsthaft auseinandersetzt, katapultiert seine Automatisierungen in eine neue Liga.

Die meisten Make-User bleiben an der Oberfläche hängen und wundern sich über langsame, fehleranfällige Workflows. Wer aber die Parallelausführung clever nutzt, Monitoring und Fehlerbehandlung integriert und seine Szenarien modular und skalierbar aufbaut, hat die Nase vorn – im wahrsten Sinne des Wortes. Parallel Processing ist kein Spielzeug, sondern das scharfe Schwert für alle, die mit make.com nicht nur basteln, sondern wirklich automatisieren wollen. Also: Mach Schluss mit dem Bottleneck-Bingo – und lass deine Workflows endlich richtig laufen.